MOBILITY AND PRIVACY: EXPLORING TECHNICAL AND SOCIAL ISSUES IN EMERGING PERVASIVE SENSOR NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Mikhail Alexandr Lisovich

May 2010

MOBILITY AND PRIVACY: EXPLORING TECHNICAL AND SOCIAL ISSUES IN EMERGING PERVASIVE SENSOR NETWORKS

Mikhail Alexandr Lisovich, Ph.D.

Cornell University 2010

This dissertation considers two important topics related to advancing wireless sensor network (WSN) technology: the privacy concerns raised by ever increasing collection personally identifying data by sensing systems, and the opportunities for synergetic function created by imbuing sensor platforms with mobility.

On the privacy side, the dissertation focuses on the collection of power consumption data in current and future demand-response systems. We build a data-gathering and behavior extraction system and conduct a small-scale monitoring experiment on a private residence. Our results show that certain personal information may be estimated with a high degree of accuracy.

On the mobility side, we consider two difficult problems in multi-agent coordination: the Multiple Path Consensus (MPC) problem, and the Multiple Sensing Region Field of Interest (MSRF) problem. We characterize both problems as NP-complete, then proceed to develop computationally tractable formulations for each. We then develop algorithms which are able to solve practically-sized instances of these problems to optimality.

Finally, we develop a practical real-world platform upon which to test multi-agent coordination algorithms, and give an example 'iteratively-deployed WSN' application.

BIOGRAPHICAL SKETCH

Mikhail Alexandr Lisovich is a PhD candidate at the School of Electrical and Computer Engineering at Cornell University, working in Prof. Stephen Wicker's WISL Networks Research Group. His current interests include emerging privacy concerns in next-generation pervasive sensing systems, particularly those associated with upcoming demand-response and SCADA technologies. They also include studying the impact of mobility on large-scale sensor networks, both in considering the network's initial deployment and using mobility to improve sensing effectiveness and network topology. Mikhail is a member of the Team for Research in Trustworthy Systems (TRUST) center, and a participant in NSF's Mobile Autonomous Systems and Technology (MAST) consortium. He has a BS in Electrical Engineering and Physics from The Pennsylvania State University.

To my parents

ACKNOWLEDGEMENTS

I would like to sincerely thank the many individuals who over the past four years have guided and supported me, as well as those talented professors, graduate students and Masters of Engineering Students alongside whom I've had the pleasure to work:

Prof. Stephen Wicker, for giving me the countless wonderful opportunities I've enjoyed over the past four years, as well as introducing me toward the privacy and legal aspects of sensor networking technology, and thus igniting my interest in patent law.

Prof. Diedre Mulligan, my collaborator on the in-home privacy work, for her excellent advise and guidance on understanding the complex legal and sociological issues involved.

Sergio Bermudez, my collaborator on the MSRF problem, Prof. Richard Karp, for his gracious help with characterizing the complexity of the MSRF problem, and Professor David Shmoys, for his insightful comments and suggestions related to the problem.

Prof. Shankar Sastry, for graciously hosting me during the summer of '08. Hoam Chung and Songhwai Oh, who guided me and collaborated with me on research during the same time.

Prof. Maxim Likhachev, with whom I've worked during the summer of 2009, and who has since collaborated closely with me to develop the MPC algorithm.

The Team for Research in Ubiquitous Secure Technologies (TRUST), both for its financial support and the opportunities it provided for academic and

My hardworking MEng Students: Devashri Trivedi, Pratik Kotkar, and Rick Wu.

My research group, who've provided invaluable advice, support, and

friendship during my time at Cornell.

Po Yan, Phoebus Chen, and all others who helped me immensely along the way, and to whom I'm deeply grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

This dissertation deals several problems relating to privacy and mobility in emerging pervasive wireless sensor networks.

A wireless sensor network (WSN) consists of spatially distributed, autonomous sensor platforms, which together monitor, analyze, and act on conditions within their environment. The individual sensor platforms (also termed 'nodes'), typically consist of one or more sensors which monitor environmental variables such as temperature, pressure, vibration, etc., a microprocessor, a radio transceiver, and a power source. In addition, there typically exist one or more base stations, which aggregate and process data from the nodes. The nodes may communicate directly with a base station or another node (single hop), or use other platforms within the WSN to relay their data (multi-hop). The nodes may be preorganized into a network by the designer, or may autonomously and opportunistically organize themselves into an ad-hoc network upon deployment. They may establish a communication hierarchy and network topology, locate themselves relative to other nodes, estimate the communication inter-node channels, map the sensing requirement, and perform any other tasks necessary to establish an effective data gathering and communication system. In addition, protocols exist for power management, interference management, data-preprocessing, and a myriad of other problems [1].

[1] For an excellent introduction to the art and science of constructing sensor networks, see Networking Wireless Sensors [32]

Wireless sensor networks are extremely versatile, and may be used for a large and constantly expanding range of applications. These applications encompass environmental monitoring, industrial production, advertising, social networking, and almost any other problem which benefits from the availability of spatially-precise, up-to-date 'real-world' data. Because of this versatility, they take on a wide variety of forms and operate under many different regimes. Accordingly, there is a wealth of work relating to all aspects of optimal WSN organization and operation [32].

In this dissertation, we focus on two important areas in WSN design - the opportunities for synergetic function created by imbuing sensor platforms with mobility, and the privacy concerns generated by advancing sensor technology.

1.2 Mobility

The sensing platforms in a WSN need not be stationary - they may have some means of navigating their environment. This additional degree of freedom makes sensor networks applicable in a large number of scenarios, including search-and-rescue, habitat monitoring, and military surveillance. It has recently received attention from both research institutions [11, 22] (See Fig. 1.1a), and governmental entities such as the Army Research Laboratory [3].

In considering mobility, our motivating example is the Smart Dust project [29], which aims to develop extremely small and inexpensive wireless nodes which can be deployed in large numbers over a particular field of interest (FoI). We believe that certain enabling technologies—such as the jumping platform being developed by the Robotic Fleas project in Berkeley [14] (See Fig. 1.1b),

(a) (b)

Figure 1.1: Enabling Technologies for WSN's: (a) shows the MIT SwarmBot project; (b) shows Berkeley's MicroFleas project.

will eventually imbue these sensors with limited mobility.

The node platforms will then be able to use their mobility to:

1. Reconfigure themselves for better connectivity or network topology
2. Fill gaps in sensing coverage
3. Share computational, power, or sensing resources
4. Mitigate interference during simultaneous transmission
5. Improve performance of cross-layer optimization
6. Dynamically respond to new tasks
7. Gracefully deal with node failures and new node additions.

The main question facing a mobile network designer is designing efficient and scalable reconfiguration algorithms which yield optimum performance or

close-to-optimum performance when filling any of the above-mentioned roles. In this thesis, we will study two complex problems in mobile agent interaction.

1.3 Privacy

In describing wireless sensor networks, the previous sections focused mainly on the technological aspects of emerging pervasive sensing systems. However, there also exist significant social issues related to WSN design. The adoption of ever more capable sening systems, as well as their increasing prevalence in every aspect of daily life raises serious ethical concerns. Without proper safeguards, the personally identifying information collected by these systems may be exploited by interested parties in ways that intrude on an individual's privacy and liberty [2].

In this dissertation, we will study the potential for exploiting the consumption data generated by upcoming demand-response systems, and the potential privacy implications of such exploitation.

1.4 Contributions and Dissertation Organization

In Chapter 1, we provided a general overview of wireless sensor networks, as well as some of the social and technological issues involved in their continued development. In particular we've reviewed the privacy implications of increasingly capable and pervasive sensing, as well as discussing the advantages that

[2]For an excellent primer on the complex array of concepts and issues involved in defining privacy, we refer the reader to Daniel Solove's Understanding Privacy [43]

mobility can offer as a degree of freedom, in terms of both sensing and network optimization.

In Chapter 2 we explore on the privacy side of WSN development. We focus on one particular representative problem - namely, the emerging privacy concerns in upcoming residential and commercial demand-response systems. Our main claim, substantiated by study results, is that in a lax regulatory environment, the detailed household consumption data gathered by advanced metering projects can and will be repurposed by interested parties to reveal personally identifying information about an individual's activities. Our contributions include the development of an 'information disclosure metric', which may be used with various theoretical frameworks to evaluate a particular technology's impact on an individual's privacy. They also include the development of an in-house-monitoring and behavior-extraction system, and the subsequent undertaking of a small-scale monitoring experiment. The experiment allows us to construct a sample disclosure metric, and establishes that certain personal information can be estimated with a high degree of accuracy.

In Chapters 3 through 5, we consider issues related to node mobility, focusing on two difficult problems in multi-agent coordination. In Chapter 3, we consider the Multiple Path Consensus (MPC) problem, wherein a number of mobile agents must navigate optimally around a cluttered environment, while satisfying time-indexed inter-agent distance constraints. We characterize the problem as NP-complete, then analyze it to come up with a heavily compressed, computationally tractable Joint State-Space (JSS) formulation. We proceed to develop an optimal, correct, and complete search algorithm for the problem, verifying its performance through simulation.

In Chapter 4, we consider the Multiple Sensing Region Field-of-Interest (FoI) (MSRF) problem, wherein agents equipped with various combinations of sensors are tasked with optimally covering an FoI having regions with various combinations of sensing requirements. We first analyze the problem to decompose it into spatial and assignment subproblems. We give a solution to the spatial problem, and characterize the assignment problem as NP-complete. We develop a mathematical programming framework for the assignment problem, then formulate a provably correct, complete, and distributed iterative search algorithm to provide an optimal solution. Having verified the algorithm's performance through simulation

In Chapter 5, we aim to build a practical real-world platform upon which to test multi-agent coordination. The work is a part of DARPA's Micro Autonomous Systems Technology (MAST) initiative, which aims to develop surveillance systems comprised of small, mobile, networked sensors. We build a demonstration wireless sensor network system, which is then used for an 'iteratively-deployed sensor network' application. Chapter 6 concludes.

CHAPTER 2

INFERRING PERSONAL INFORMATION IN UPCOMING DEMAND RESPONSE SYSTEMS

2.1 Introduction

A radical transformation of our nation's power distribution systems is well underway. Next generation Supervisory Control and Data Acquisition (NG-SCADA) architectures, now under development, will precipitate an exponential increase in both the collection of data and extent of control available to consumers and utilities. Utilities are increasingly adopting automated metering, advanced demand response architectures, microgrids, and other systems which will provide cost savings in power generation, increase grid reliability and flexibility, and create new modes of consumer-utility interaction.

Several pilot microgrid projects [1] have been deployed in recent years; there has also been increased deployment of Advanced Metering Infrastructure (AMI) systems by major utilities across the US. According to a 2008 Federal Energy Regulatory Commission staff report [5], five percent of meters installed in the US are 'smart' meters, and eight percent of U.S. customers are participating in demand-response programs. Furthermore, the smart grid has recently become a presidential priority, receiving 4.5 billion dollars that must be spent on its deployment within the next two years. This infusion of funding means

The work in this chapter was done in collaboration with Diedre Mulligan, who is a professor at the Berkeley School of Information, and Stephen Wicker, who is a professor at the Cornell School of Electrical and Computer Engineering.

that market penetration for both smart meters and demand response is likely to increase dramatically in the near term.

Next generation SCADA projects may benefit utilities, consumers, and new market players. For the power companies, automated metering will reduce the costs of data collection and improve large-scale load planning and long-term research through real-time feeds of energy consumption. This research will allow utilities to improve planning and test the effects of various demand side management programs. For the consumer, the projects will result in potentially lower costs, more information about consumption patterns, more control over power use, and the ability to actively participate in power generation . In addition, increased knowledge and management tools may reduce overall consumption. The engine for these activities- per-household consumption data- poses both privacy and security risks.

This article is part of a larger effort by the TRUST[1] NSF Science and Technology Center to promote a robust, secure, and trustworthy smart-grid. Our teams focus on the confluence of sensor networking, power distribution, and policy in order to address the privacy and security issues that emerge from a substantial increase in power system monitoring at the consumer level. Our claim in this article (we will refer to it as the *main claim*) is that in the present regulatory and judicial environment, it is both possible and probable that the household consumption data gathered by advanced metering projects will be repurposed by interested parties to reveal and exploit personally identifying information about the programs' participants.

The smart grid is bringing new, nontraditional players into the energy con-

[1]TRUST is a multi-university NSF Science and Technology Center focused on trustworthy systems. See [4].

sumption market, many of them not covered by existing privacy and security regulations. The real-time nature of the data flows and their increasing granularity will generate new interest in access and reuse by these players, which include law enforcement, marketing, as well as nefarious individuals.

Without proper technical, procedural and legal safeguards, access to detailed household consumption data raises ethical concerns. It could be used to facilitate burglary, to initiate targeted advertising based on activities occurring wholly within the home, and in the case of law enforcement, monitor home-based activities in real-time.

While the sanctity of the home holds a special place among the Constitutional privacy protections in the U.S., the capture and storage by businesses of information about private activities have eroded the protections afforded to it. In United States v. Miller, the Supreme Court held that individuals have no reasonable expectation of privacy in data voluntarily given to and held by third parties. Since this ruling, several state constitutions have been interpreted to provide some privacy protection for information about consumers captured in business records, and. Additionally, a leading U.S. Supreme Court case interpreted the Fourth Amendment to require law enforcement to obtain a warrant prior to aiming a thermal imaging device at a residence, since it revealed detailed information about occupants' activities. However, it is unclear what if any constitutional limitations will apply to the access and use of these detailed energy records by the government. Regardless, robust privacy protections are best produced through a mix of technology, regulations aimed at the private sector, and regulations - hopefully codified as constitutional limitations - on government use.

The importance of this issue has recently been noted. A 2009 report by the National Institute for Standards and Technology (NIST) [8] highlights the need for privacy to be attended to during the development of smart grid standards and implementation plans. In addition, a recent ruling by the California Public Utilities Commission [17] grants consumers the right to control many of the uses and disclosures of household level consumption data. The technical design of and policies for the smart grid, being worked out today in the U.S. and in other places around the world, will have a lasting impact on the privacy of in-home behavior. Consequently, there is a need for a sustained and thoughtful research-based approach to establishing appropriate technical and legal protections.

Discussion and advocacy efforts are already underway. Lerner and Mulligan have written an article [34] chronicling the evolution of court opinion toward energy data privacy and calling for its constitutional protection. They have also collaborated with the California Public Utilities Commission (CPUC) to develop a set of draft guidelines [30] for a secure and privacy-preserving demand response infrastructure.

In this body of work, we contribute to the discussion by exploring technical aspects of the main claim, focusing on data generated, what it reveals and how and why it may be collected and repurposed[2]. Our contributions include highlighting the importance of certain algorithms for extrapolating activity information from power consumption data, a formal way of evaluating information disclosure, and an illustrative proof of concept technical study.

The rest of this chapter is concerned with systematically developing and

[2]Readers interested in the long-term legal and privacy implications of in-home monitoring are encouraged to read our colleagues' paper [34] published in the Stanford Technology Law Review

substantiating certain aspects of our claim. In Section 2.2, we familiarize the reader with the current state of advanced metering technology. We also describe Non-Intrusive Load Monitoring (NILM) systems and algorithms, singling them out as a fundamental tool for extrapolating activity. In Section 2.3, we discuss some of the parties interested in the data and their motivations for obtaining and repurposing it. In Section 2.4, we aim to help in formalizing these parties' impact on individual privacy by discussing an information disclosure metric that quantifies the ways that privacy can be infringed. In Section 2.5, we prove that repurposing is feasible from a technical standpoint by conducting a small-scale monitoring experiment on a private residence. Our results show that, even with relatively unsophisticated hardware and algorithms, occupant activity can be estimated with a high degree of accuracy. In Section 2.6 **we point out the experiment's limitations**, and discuss how our experimental methods can be extended to larger scales. Finally, in Section 2.7 we summarize data-handling guidelines suggested by other TRUST researchers and discuss how our findings fit into the ongoing discussion.

2.2 Technology Overview

To familiarize the reader with the technical aspects of the issue, we begin with a brief overview of demand response technologies[3].

[3]For a more complete overview of AMI and NILM, we refer the reader to [27] and [33], respectively.

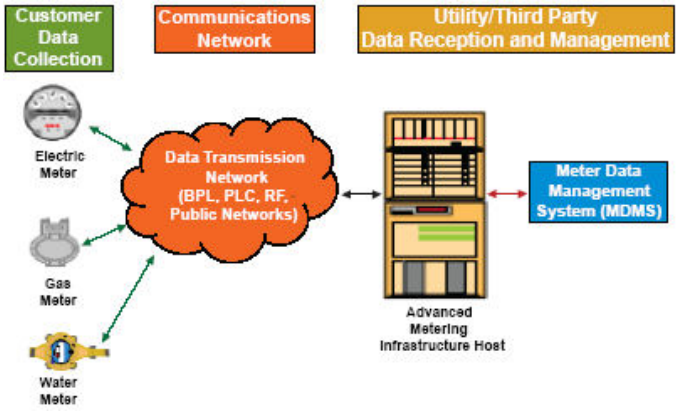


Figure 2.1: The building blocks of an AMI system. Note: Figure taken directly from [27]

2.2.1 Advanced Metering

In a typical Advanced Metering setup (See Fig. 2.1), the customer is equipped with solid state electronic meters that collect time-based consumption data at daily, hourly or sub-hourly intervals. These meters then transmit the collected data to the Meter Data Management System (MDMS) that manages data storage and analysis, shaping the information into a form useful for the utility [27].

As mentioned in the introduction, AMI systems have already been deployed in large numbers. The reader is referred to [5] for detailed statistics on deployment and system capabilities.

2.2.2 Non-Intrusive Load Monitoring

A NILM system collects data much like its AMI counterpart, but goes a step further by processing the data to determine the operating schedules of individual electrical loads. This is typically done by disaggregating the collected data stream into individual load signatures and matching each signature with refer-

ence signatures stored in a database. For private residences, these loads are usually appliances such as the refrigerator, air conditioner, or water heater. These systems are used for a wide variety of purposes, including load research and implementation of incentive programs for particular appliance usage patterns [2].

Current NILM systems require data with a second/sub-second resolution. Because of this, processing is usually done locally, at the electricity meter. However, it is possible to run these algorithms remotely, and useful results may be obtained even with the sparse data provided by an AMI system. Therefore, when considering how power consumption data can be repurposed and the kinds of information that can be extracted from it, one should consider a NILM algorithm as an essential building block. We will develop this thought in Section 2.4.

2.3 Players, Use Cases and Motivations

Utilities typically have policies that provide protection for utility records and personal information. For example the California Energy Commission requires the consumer's written consent for the release of personal data related to billing, credit, and power usage [39]. Utility records may be released in certain circumstances if the customer is not identified, and exceptions are made for law enforcement access.

Given these policies, there exist agencies, organizations and individuals who have natural motives to use power consumption data for purposes other than load research and demand response.

2.3.1 Law Enforcement Agencies

Federal and state law enforcement agencies currently access utility records for a range of purposes. They are aided by current jurisprudence,[4] which allows easy access to public utility records and provides legal precedent for their use in prosecuting criminal cases.

Police routinely use public utility records to seek out drug producers. KXAN Austin recently reported that the Austin Police Department has an agreement that allows it to access Austin Energy power usage records without a search warrant [40]. Investigators have used their access to screen consumers for possible drug production, relying on the fact the heat lamps and watering systems used to grow marijuana indoors can increase a consumer's energy consumption far beyond the norm. While Austin appears today to be an exceptional case since many utilities require a subpoena for releasing records, the program hints at the growth in use we might expect as precedent increasingly detailed consumption data becomes available. purposes. As more granular consumption data begins to flow to utilities in real-time, law enforcement interest in it is quite likely to grow.

2.3.2 Marketing Partners

Behavior and appliance usage information may potentially be used for directed advertisements. For example, some NILM systems are powerful enough to identify specific appliance brands, and may even identify malfunctioning appliances [33]. A marketing company partnering with a utility may use this data to

[4]The reader is referred to the Stanford Law article[34] for a more in-depth discussion.

send customers targeted advertisements for repair/upgrade, or more generally derive demographic data for broader advertising claims. Targeted advertising based on in-home activities transgresses the current norms of information flow and create new privacy concerns. The exposure of in-home activities and the resulting marketing may meet with strong consumer disapproval.

2.3.3 Criminals

In their article [34], our TRUST colleagues give an excellent scenario for criminal abuse of power consumption data: criminals could tap into an intermediate AMI node or simply monitor the unencrypted traffic between it and the individual meters. They could process the data to compile lists of household appliances, or to determine occupancy patterns of houses in the entire neighborhood. Knowledge of occupancy patterns would facilitate burglary or some other property crime, while appliance lists will help with choosing targets.

2.4 Quantifying Information Disclosure

The previous section showed by way of examples that the evolution of monitoring technology creates new risks to individual privacy by exposing data previously held within the home to a mix of parties. However, it is not apparent just how these risks can be quantified, especially as a function of available data. There is a need for a 'privacy metric,' that associates the degree of data availability with potential privacy risks. Evaluating these risks is a complex process that must necessarily take into account common industry practices regarding

data privacy, the state of current jurisprudence, the consumers' expectations of privacy, and the relationship between utilities and interested parties. While we feel that such an analysis is not within our purview, we believe that we can provide a crucial technical component for the analysis – a 'disclosure metric,' which associates data quality (accuracy of readings, time resolution, types of readings, etc) from a particular source with the information that may potentially be revealed by the data.

To construct this disclosure metric, we need to better understand the nature of the information that can be extracted from available sensor data. Thus, we will start by suggesting a formal framework for extrapolating activity, then use it to construct our metric. **We will also suggest various privacy-theoretic frameworks that can be used in conjunction with the disclosure metric to move towards a robust privacy metric.**

2.4.1 Extrapolating Activity

Extrapolating activity may be thought of in two stages - during the first 'intermediate' stage, NILM in combination with data from other sensors is used to extract appliance usage, track an individual's position, and match particular individuals to particular observed events. During the second stage, the intermediate data is combined with demographic data, such as the number/age/sex of individuals in the residence, tax and income records, and models of typical human behavior. Together, these data are used to identify activities, behaviors, preferences, and so on. The two stages are not cleanly separated - raw data may be used directly to estimate a parameter of interest, and determination of some

intermediate parameters may rely on contextual information. However, many parameters in the second stage rely on the same intermediate data (e.g. sleeping habits and eating habits may both be extrapolated from tracking data.)

The first stage is more readily quantifiable - for a particular algorithm, parameters take definite form (such as use/condition of appliances, tracking), while performance can be evaluated in statistical or information-theoretic terms. However, it is more difficult to define an absolute performance limit for the second stage - the number of specific preferences and beliefs that can be estimated is virtually limitless. In order to develop a comprehensive disclosure metric, one needs to carefully define a list of 'important' parameters, basing importance both on how fundamental a parameter is (how many other parameters may be derived from it) and on home/business owners' expectations of privacy. Expectations of privacy, in turn, are partially based on previous incidents of abuse and repurposing (such as the one in Section 2.3.1). The list of second stage parameters may be hierarchical, with more specific parameters being used to evaluate more general ones. Once an appropriate list is defined and 'importance' values assigned, it is possible to determine the sufficiency of available data based on requirements of current and future NILM, tracking, and other relevant algorithms.

A list of important second-stage parameters establish the evaluation criteria. Algorithms for estimating the parameters, along with the corresponding data requirements, provide a method for evaluating the sufficiency of the available data. Together, these provide a metric for how much information may potentially be revealed by a particular monitoring system.

Such a metric does not aspire to be an absolute measure, in the sense of

encompassing all existing disclosure scenarios or anticipating all future ones. However, if thoughtfully implemented, it provides a valuable tool for evaluating privacy risks associated with a particular system, as well as allowing for comparison between similar systems. We will construct a sample disclosure metric in Section 2.5.7.

2.4.2 Using the Disclosure Metric to Assess Privacy Loss

The disclosure metric metric can be used in conjunction with one of several theories of privacy to assess the actual privacy loss. For example, Helen Nissenbaum's theory of privacy as contextual integrity [23] can be used to examine how the data loss of the new system relates to context dependent norms of information appropriateness and flow. One could also use the principles laid out in Surden's Structural Privacy Rights essay [24], along with Lessig's 'What Things Regulate' chapter in Code 2.0 [35] to explain how the new flow of data removes a structure that afforded privacy protection for in-home activities (i.e. the walls, combined with low level of detail about energy consumption), and replaces it with a system that breaches the walls of the home and exposes real-time consumption data.

2.5 Experiment

Although it is known that first-stage parameters such as appliance usage may be accurately estimated (see performance chart in [2]), and repurposing of sensor data has been previously explored [28], to our knowledge, our group is the first

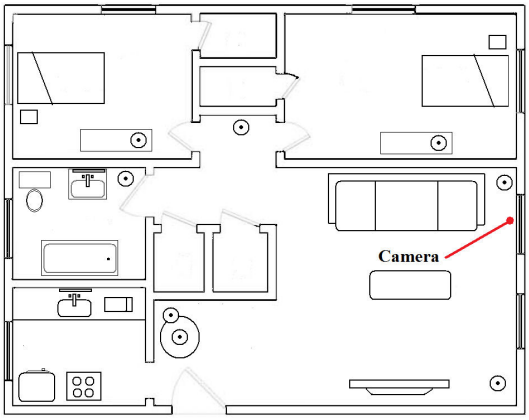


Figure 2.2: The floorplan of the small student residence used in the experiment.

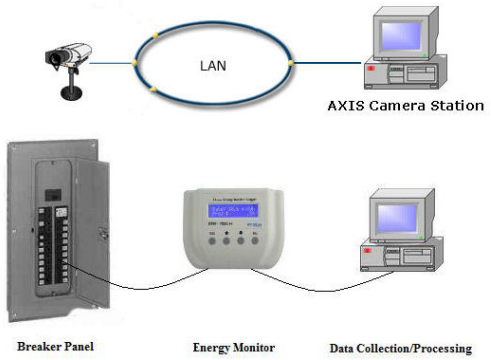


Figure 2.3: Camera and electrical data gathering setups.

to attempt extrapolating activity from power consumption data. In this work we want to prove that activity extrapolation is feasible, thus lending credibility to our main claim and providing an experimental precedent which others can cite in future efforts. To do this, we conducted a small-scale monitoring experiment on a private residence.

2.5.1 Experimental Setup

We conducted our experiment in a typical student residence (Figure 2.4.2). For data gathering, we used the Brultech EML energy usage monitor. Figure 2.4.2

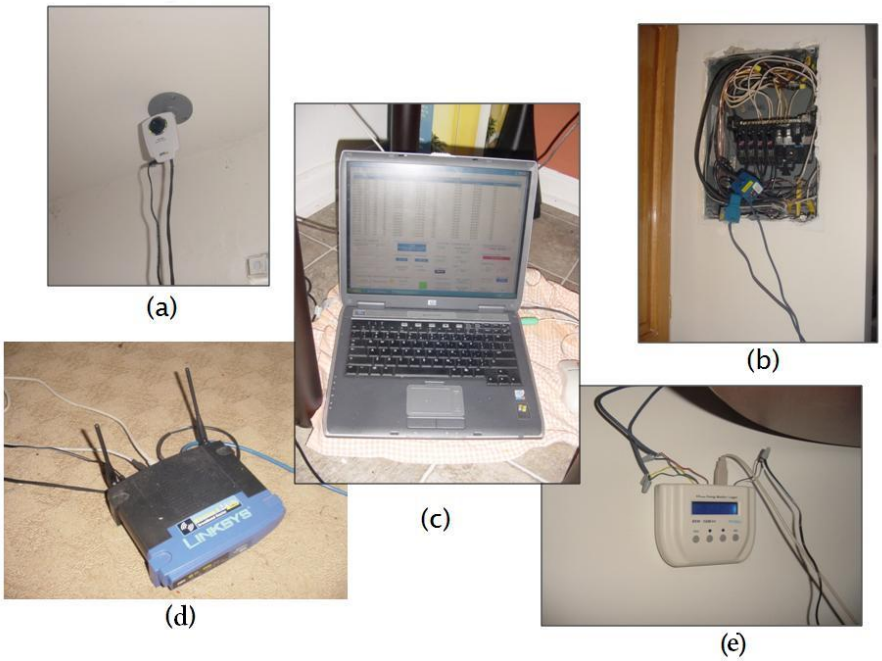


Figure 2.4: Sytem Setup: (a) shows the axis camera used to gather reference activity data; (b) shows the current sensors attached to the breaker panel and leading to the energy monitor; (c) shows a PC which was used to record the camera and electrical data; (d) shows the router used to connect the camera to the PC; (e) shows the energy monitor.

shows the data gathering setup. The energy monitor was attached to the residence's breaker panel and sent real-time power usage information to a workstation responsible for data collection. The station recorded power usage at intervals of 1 or 15 second(s) and with a resolution of 1 Watt. The same workstation then ran the NILM and behavior extraction algorithms. To evaluate the system's performance, we placed a network of cameras around the residence. We elected to use the Axis 206 network camera (position shown in Figure 2.4.2), which we connected to a workstation using an Ethernet switch. The workstation ran the AXIS Camera Station software and recorded motion events for later processing. The camera control setup is shown in Figure 2.4.2. Photos of the system as deployed within the residence are given in 2.5.1

2.5.2 Experimental Protocols

The experiment was run semi-continuously over a period of two weeks. This timeframe allowed us to obtain repeated data for pattern matching while accounting for time constraints. Power and camera data collection software was shut down on a semi-daily basis for archiving, maintenance, and manual video data processing.

Electrical data was collected from the house breaker and passed to our behavior extraction algorithm through a bridge program. Camera data was collected by the Axis Camera Station software and stored in MPEG format at a resolution of 320x240 at 4 fps. At regular intervals, video data was manually analyzed and processed into activity logs. Upon completion of the logs, the original video data was deleted. Activity logs had the following format:

Date/Time Subject Activity

The subject could be any of the house's three residents or a guest. Possible activities included turning any of the household appliances on or off (ex: *kitchen_lamp_1_on*), entering or leaving the residence, sleeping, preparing meals, taking a bath, or having a party. Note that because the cameras were not put in individual rooms, the resulting activity logs were not fully complete. However, this arrangement respected the residents' privacy and lead to more natural behavior in areas under visual observation, while the collected data were sufficient to estimate parameters of interest (see Section 2.5.4 for the parameters).

After collection, the experiment data was subdivided into two sets: a smaller three day 'Training' set and a larger seven day 'Experimental' set. While we ac-

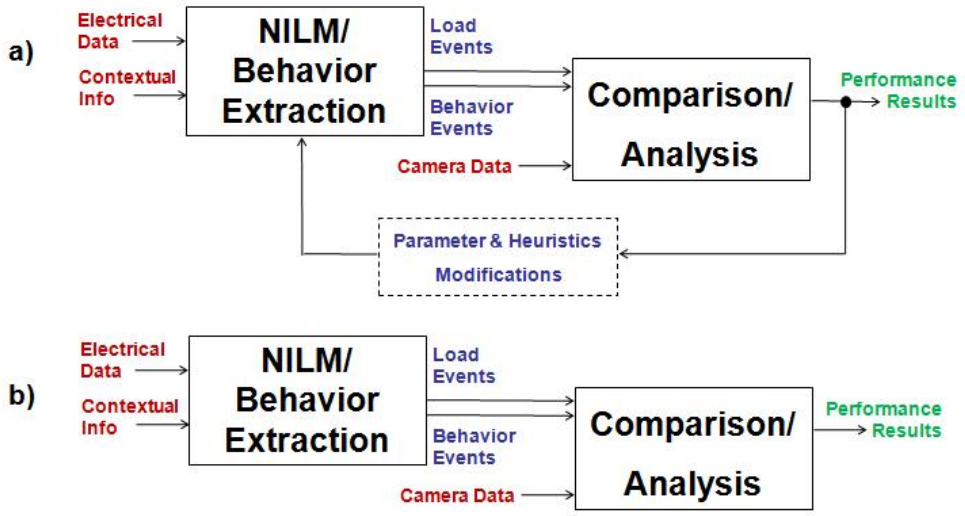


Figure 2.5: Flow of information between experiment components. Note that the parameter modification loop in (a), used during training, was removed for the experiment (as shown in (b)).

tively modified our algorithms to increase performance on the Training set, we kept them completely unchanged on the Experimental set. Figure 2.5 shows the information flow between various components of the experiment for both the training and experimental stages. Please refer to it as you read the subsequent sections.

2.5.3 Threat Model

For the purposes of this experiment, we assume an adversary that has access to Real power data from a single household at a power resolution of one Watt and a time resolution of at least fifteen seconds. We further assume that the adversary has a list of appliances present inside, as well as their turn-on/turn-off profiles **(not an unreasonable assumption – the Enetics NILM system [2] has a built-in library of generic appliance profiles, which can be matched to**

unknown load signatures). Additionally, we assume that the adversary can distinguish between intermittent and periodic loads. This information need not be manually obtained - a list of intermittent appliances can be compiled with the aid of reference software such as [2] or through automated means [13], and periodic loads can be automatically identified with existing NILM algorithms [12].

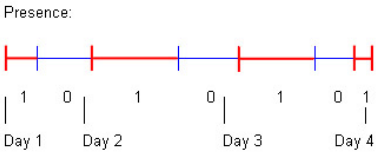
2.5.4 Parameters to be Estimated

We chose several parameters that were both revealing and possible to estimate using our data gathering equipment and processing algorithms. They are:

- Presence/Absence - whether or not someone is present at the house
- Appliance Use - microwave, stove, water heater, TV, misc appliances, etc.
- Sleep/wake cycle - when do the household's occupants wake up and fall asleep.
- Other Significant Events - Breakfast, Dinner, Shower, Party, etc.

More formally, we begin by combining all data into a single timeline. For each parameter, we partition this timeline into segments, with each segment assigned some value. For most parameters, the value is binary, indicating whether a person is present or absent, asleep or awake, etc. For a specific parameter, the i^{th} 'on' interval is defined by T_i^{on} and T_i^{off}.

An example partition for the Presence/Absence event is shown below:

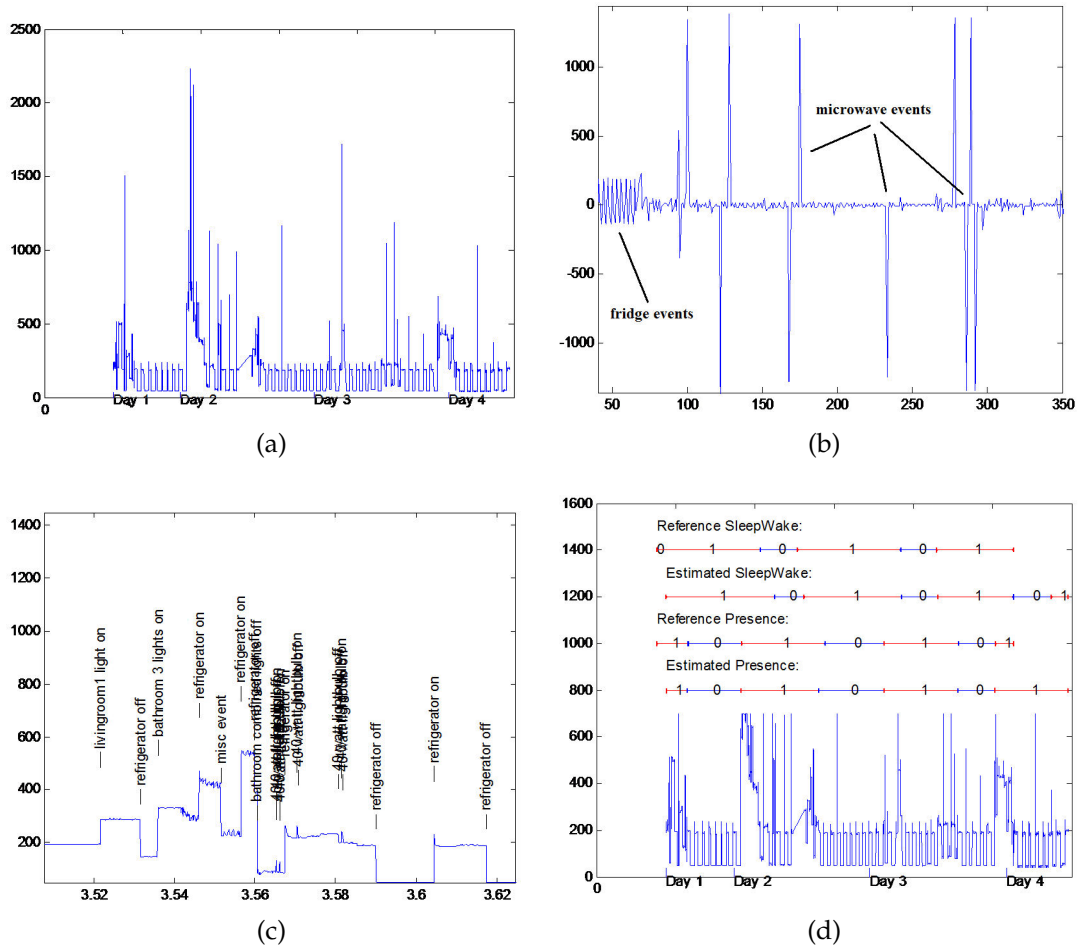


2.5.5 Performance Metrics and Evaluation

Once energy use data is gathered and processed with behavior extraction algorithms, we wish to compare the results against reference results obtained from camera data. To do this, we employ two classes of metrics. The first class is event-based and consists of the Failure-to-Detect/ Misdetection percentages for each parameter. These percentages are computed by using the following procedure:

1. Define the cutoff threshold T_{thresh}, choosing it based on experimentation with training data
2. For each parameter, examine the sequence of turn-on/turn-off events on both the reference and estimated intervals.
3. If a camera event occurs but a corresponding electrical event does not occur within T_{thresh} seconds, declare a *Failure to Detect*.
4. If an electrical event occurs but a corresponding camera event does not occur within T_{thresh} seconds, declare a *Misdetection*.

The second class of metrics takes a broader perspective by computing the percentage of the reference interval that is correctly classified. This may in some cases be a better indicator of long-term performance, since the algorithm may miss several short-duration events while classifying the vast majority of the interval correctly.



(a) (b)

(c) (d)

Figure 2.6: Behavoir Extraction Algorithm: (a) shows the aggregate power consumption data; (b) shows the derived switch events; (c) presents several identified load events; (d) compares reference and estimated intervals.

Together these metrics help one get a well-rounded picture of the algorithm's performance, providing both detail and global perspective.

2.5.6 Behavior Extraction Algorithms

Our behavior extraction system is implemented in MATLAB and consists of two major components: a NILM algorithm and a suite of functions that estimate the high-level parameters mentioned in the previous section. The NILM algorithm

Table 2.1: Appliance List

Appliance	Turn-on/Tur-off Char.
Bathroom 3 Lights	181 W
Bathroom Fan & Light	126 W
40 Watt Lightbulbs	40 W
Livingroom Light 1	20 W
Livingroom Light 2	40 W
Passageway Light	55 W
Microwave	1000 W
Refrigerator	$+\Delta$250W,$-\Delta$110W/140 W

we implemented is based on an early MIT prototype [19]. It analyzes the electrical data (Fig. 2.6a) gathered by the load monitor, performing edge detection and cluster matching.

During edge detection, the algorithm computes a difference series $\Delta(t) = P(t) - P(t-1)$ from the electrical data $P(t)$. Adjacent $\Delta(t)$'s of the same sign and greater than a certain threshold are merged into *switch events* (Fig. 2.6b).

During cluster matching, switch events are matched against a database of load signatures and classified as either 'on' or 'off' events. A load signature may be a switch event of a certain magnitude (a 40-watt light bulb has a step turn-on signature of $\Delta(t) = 40$ Watts) or a series of such events (a refrigerator has a turn-on signature of $\Delta(t) = 1100$ W, $\Delta(t+1) = -960$ W). Unclassified events are either discarded as noise or labeled with a catchall 'misc. event' classifier. The loads to be detected for our residents, along with their turn-on/turn-off signatures, are given in Table 2.1. A sample of classified events is shown in Fig. 2.6c.

During anomaly resolution, the algorithm tries to classify the miscellaneous events as a combination of different turn-on/turn-off events. This allows for

classification of events that occur close to each other.

Once the load events are classified, behavior extraction routines use them to determine presence schedules, sleeping cycles, shower and bathroom use, mealtimes, and other activities. We briefly describe the most important routines:

- Presence - Because the refrigerator is the only load in the residence with automated turn-on/turn-off events, we assume that *any* non-refrigerator event indicates presence. On the other hand, absence is defined by low power usage and lack of events. An extended interval with low power usage during which no events occur implies that all subjects have left the residence.
- Sleep Cycle - Intervals of inactivity occurring between late evening and early morning are likely to imply that all people are sleeping (as opposed to absent). Therefore, all such absence intervals are reclassified as sleep intervals.

The last major component of our system is the analysis suite. Reference data derived from camera logs is automatically processed into reference intervals, which are then compared against estimated intervals using metrics described in Section 2.5.5. Sample output, showing reference and estimated intervals for both presence and sleep cycles, is shown in Figure 2.6d.

As a note, we want to highlight that activity extrapolation is quite feasible in real time. A modified version of our system (not used in our experiment) can process electrical data, extract load events, accurately classify them, and update behaviors of interest in lockstep with arriving data. Figure 2.5.6 shows this system, which consists a scrolling chart, to which classified load events are

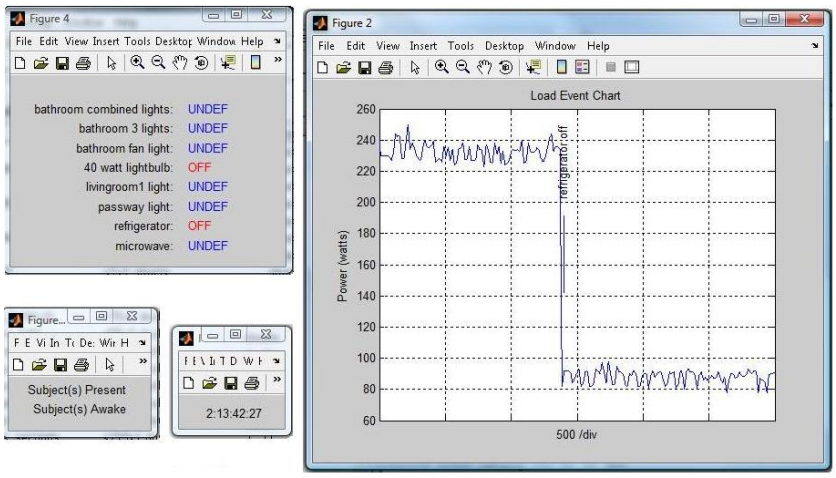


Figure 2.7: Behavoir extraction algorithm GUI

appended in real time. Current state of appliances is shown in the top right window, while the current state of the apartments' occupants (Present, Awake) is shown in the bottom right.

2.5.7 Results

Our algorithms were run on two sets of data: a smaller three day 'Training' set and a larger seven day 'Experimental' set. As mentioned previously, while we actively modified the algorithms to increase performance on the Training set, we kept them completely unchanged on the Experimental set. The results are shown in Table 2.2. For each quantity, the table's second column gives the number of events recorded, the third column gives the percentage of successfully detected reference events, the fourth column gives the misdetection percentage, and the fifth column states the percentage of reference interval correctly classified.

One important appliance left out of Table 2.2 is the refrigerator, which au-

Table 2.2: Algorithm Performance

	Sample Size	Ref. Events Detected	% Misdetects	% Int. Correct
Training Data				
Presence	8 Ref., 8 Est.	100%	0%	97.3%
Sleep Cycle	6 Ref., 6 Est.	100%	0%	93.4%
Microwave	8 Ref., 8 Est.	50%	78%	43%
Bathroom Lights	8 Ref., 8 Est.	72%	44%	52%
Passage Light	8 Ref., 82 Est.	38%	90%	57%
Living Room Lights	8 Ref., 8 Est.	55%	88%	58%
Experimental Data				
Presence	10 Ref., 10 Est.	80%	20%	97.4%
Sleep Cycle	12 Ref., 10 Est.	83%	0%	92.3%
Microwave	10 Ref., 58 Est.	80%	83%	99%
Bathroom Lights	60 Ref., 103 Est.	63%	42%	81%
Passage Light	8 Ref., 82 Est.	38%	90%	57%
Living Room Lights	19 Ref., 179 Est.	21%	89%	52%

tonomously cycles between high and low states. Unfortunately, we did not observe these state transitions directly - this would have required a separate energy monitor exclusively for the refrigerator. However, we can comment on the algorithm's performance by manually examining the electrical data readout (a refrigerator has a distinctive operating profile - see Figure 2.6c). For the training data set, 101 of approximately 104 refrigerator events (more than 97%) were correctly classified. The success rate was similarly high for the experimental data set.

Generally, the algorithm performed quite well in determining presence and sleep cycles. In both cases, over 90% of the total interval length was correctly classified, for both training and experimental data. We believe this is due to our success in identifying the refrigerator load, the small number of autonomous appliances in the residence, and the consequent simplicity of presence / sleep-wake heuristics.

The algorithm was also relatively successful with classifying microwave and bathroom light events, supporting our hypothesis that both mealtimes and shower times can be predicted with moderate success.

Detection of Living room lights was unreliable, partly due to the presence of other 40 Watt light bulbs in the residence.

It's worth noting the high percentage of misdetections. We believe this is caused in equal part by the limited capabilities of our data gathering system / algorithms, and by the imperfections of our camera monitoring setup. On the behavior extraction side, our data logger recorded only real power and only at 15 sec intervals, while our algorithm was tilted toward 'false alarm' rather than 'failure to detect'. On the camera side, our camera was not in a position to observe all loads directly, which meant that turn-on/turn-off events were sometimes missed during manual processing. Consequently, for appliances, the percentage of reference points detected is the most credible measure of algorithm performance.

Also, we note that for appliances, 'percent interval correctly classified' is not necessarily meaningful, since the 'zero-performance point', defined as the performance when the entire estimated interval is set to '0', is still 50% for microwave and bathroom lights.

2.5.8 Degree of Disclosure

Figure 4 shows an implementation of the disclosure metric whose construction we discussed in Section 2.4. We select eight important parameters that may

be revealed by our NILM-based behavior extraction algorithm, then rate the amount of information disclosure for each as Negligible, Slight Fair, High, or Severe. Where possible and appropriate, we also provide a numerical measure from one to one hundred.

We begin with Aggregate Presence and Sleep Schedules - these are useful to a broad range of players (See Section 2.3) and their performance transfers directly from Table 2.2. Based on arguments made in Section 2.5.5 for balancing the short and long-term perspectives, we average performances in columns three and five. The resulting performance numbers - over 90% in both cases, seem quite good. However, the number of people present/asleep currently cannot be tracked, which detracts from the performance somewhat. We rate the degree of disclosure for both as High.

However, the algorithm's tracking ability is only Fair - occupants can be localized to the kitchen, or bathroom, but individual movement within the house cannot be reliably localized.

Without the prior behavioral profiles and ancillary sensor data, the algorithm's ability to assign events to individuals is practically nonexistent.

For appliances and appliance derived parameters, we calculate performance by averaging all relevant entries in column three of Table 2.2, and subtracting an average of entries in column four weighed by .2 (as Misdetects are indicative but unreliable for reasons listed in Section 2.5.7) The overall ability to identify appliances is only Fair, due to both the fairly low detection rate and the high percentage of misdetects. However, the ability to identify large, distinctive appliances - in our case the refrigerator and the microwave, is High.

Figure 2.8: Degree of Disclosure

Disclosure about meal times, which we obtain by windowing the microwave use detection/misdetection ratio with likely timeframes for meals, is Fair. Information about shower times, derived directly from bathroom light & fan use, is also Fair.

Overall disclosure is the weighted (in our case equally) average of all of the above parameters - we qualify it as Fair.

The overall threat to individual privacy may now be evaluated by taking these results on potential disclosure from consumption data, qualifying them with the likelihood and degree of disclosure, historical precedent for repurposing, and the relationship between the data holders (utilities) and interested parties. While do not presume to conduct such an analysis within this article,

2.6 Discussion

Our experiment shows that presence events and sleep cycles can be estimated with high confidence, at least for a household with few appliances and relatively infrequent switching events. However, while the experiment is illustrative of the system's potential, it does not comprehensively characterize its capabilities. First, the scale of the experiment - a week's worth of data from a single residence - is too small to draw conclusions on the system's limitations. Conversely, the adversarial model may also be too strong - in practice, an adversary may not have an actual list of the appliances inside a home, which would lead to less reliable presence and sleep predictions.

However, despite these caveats, we believe that our results are sound, and that moreover there is potential for significant refinement of our approach. First, we note that the residence did not have an electric stove or a water boiler - two readily identifiable loads whose 'on' intervals directly correspond to mealtimes, laundry, and showers. Second, we have used only electrical data - a behavior extraction algorithm can combine data streams from electric, water, gas, humidity, and any other available sensors. Third, our data resolution (15 seconds in most cases) was relatively low and our behavior extraction algorithms were relatively unsophisticated, as our aim was to prove feasibility and not to optimize performance. NILM and behavior extraction systems of the near future will surely surpass our effort in performance, enabling person-to-event assignments and perhaps even limited tracking.

On the other hand, we believe that useful data can be extracted by less potent technology. Hourly power averages such as the ones produced by California's

AMI system may also be used to determine presence and sleep cycles (although to a coarser degree). Major appliances with substantial steady-state power consumption (e.g. heat lamps) can also be identified.

We note that future concerns are not limited to the performance of these systems on the level of an individual household. Because the algorithms are fully automated, analysis may be done on extremely large scales, involving hundreds or thousands of residences. Easy access to information will inevitably generate a market for it.

2.7 Guidelines

A report recently submitted to the California Energy Commission[30] makes several recommendations for power-data handling. They recommend:

1. Multiple tiers of control and oversight, both by the utilities themselves and the state/federal government.
2. Explicit guidelines regulating access to data for customer service, load research, and other functions.
3. Strong user control over information leaving the residence.
4. Protocols which do most of the data processing at stations located inside the residence, as well hard prohibitions against relaying certain types of data.

One of the authors' main points is that data mining of hourly usage data should be carefully regulated. The authors advise that more stringent rules on

the use, release and re-use of energy consumption data should be adopted as data mining practices develop and new information in which consumers have a reasonable expectation of privacy is exposed.

Our work details the sorts of conclusions that can be readily drawn from power consumption data. Our discussion of interested entities and motivations shows that the decrease in the time interval between readings of energy consumption-likely to real or near-real-time will create new interest in repurposing consumption data. Our technology discussion and proof of concept demonstration show that even the simplest data mining and pattern matching tools can convert power consumption data into information about events within "the sacred precincts of private and domestic life," [34] illustrating the extent to which residential privacy may be violated through the collection and use of power consumption data. Finally, the disclosure metric we propose and implement facilitates the evaluation of privacy risks, allowing one to more precisely define the permitted and prohibited uses of data mining.

CHAPTER 3

PRACTICAL JOINT STATE-SPACE PLANNING FOR THE MULTIPLE PATH CONSENSUS PROBLEM

3.1 Introduction

In this chapter, we investigate the problem of multiple agents - robots, or sensors imbued with a degree of mobility, navigating together in a cluttered environment. The environment is represented by a graph, and each agent has a fixed starting and destination (s-d) coordinates. In addition, there exist P time-parameterized distance constraints between two or more agents. For example, agents may need to periodically approach each other to within communication range, or conversely move sufficiently far away to mitigate interference during simultaneous wireless transmission. We seek an algorithm which, given an environment graph, a set of agents, a set of s-d coordinates, and a set of time-parameterized constraints, generates a set of paths which satisfy these constraints while minimizing some cost function [1].

An algorithm of this sort has a wide variety of applications, which we broadly categorize as either resource-sharing or interference management. Resource sharing is beneficial during joint exploration. For example, a team of agents exploring unknown territory may be required to approach each other at regular intervals to exchange mapping information. It may also be beneficial

The work described in this chapter was done in collaboration with Dr. Maxim Likhachev, who is a professor of Electrical Engineering at the University of Pennsylvania.

[1] A cost function can include time, distance traveled, power consumed, etc., and can take a linear, convex, or nonconvex form.

for transmission - individual agents may come together to share battery power for better range, network lifetime optimization, etc., share computing resources to perform data fusion, or act as a coupled multi-antenna unit. If specialization is present - i.e. one agent has a long range radio, or a substantially greater power reserve, it may rendezvous with less capable agents in order to obtain their sensor data and transmit it to an aggregation point.

With interference management, the requirements are reversed. Suppose the mobile agents have designated patrol routes, and there exists a transmission schedule, generated by some other algorithm, that the nodes must adhere to. When two or more agents transmit at the same time, they must furthermore not excessively interfere with one another. If M agents must transmit at the same time, one may generate a set of 'allowable' M-tuples (corresponding to the M positions), as well as a set of 'prohibited' M-tuples. An MPC algorithm may then be applied to generate a modified set of satisfying patrol routes. In this chapter, we will only consider maximum-distance type constraints (corresponding to resource sharing). However, in Section 3.2 we show that that the algorithm can handle other types of constraints with minimal modification.

There has been a great deal of work done on routing individual agents from source to destination via discrete graph searches. Dijkstra's seminal paper provided a simple polynomial time algorithm [18] to plan shortest s-d paths. Subsequent work has introduced countless useful variations, such as heuristic-assisted search [41], dynamic replanning [44], planning with time constraints [31] [25] [38], and accounting for different cost functions and environment constraints [37].

When multiple agents must navigate the same discrete environment inde-

pendently, the above algorithms may be used without alteration, and computation time scales linearly with the number of agents. However, when coupling constraints are introduced, the problem becomes considerably harder. Constrained multiple shortest path problems have been shown to be NP-complete for even a single constraint ([21], p. 13), and exhaustive searches in the joint state-space become intractable for all but the simplest problems. Several papers [9] [10] have dealt with distnace constrained multi-agent navigation. However, these efforts focused on maintaining connectivity, rather than satisfying time-parametrized distance constraints. Two recent works [36] [15] have addressed the MPC problem. The approach in [36] (and its extension in [15]) employs shortest path searches on N graphs whose edge costs are augmented by penalty functions, iteratively adjusting the penalty functions to guarantee eventual convergence to optimal solutions under certain conditions. While the method has been shown to converge in minutes for practical problem instances, we believe that there is potential for both significant performance improvements and a widening of the class of problems that can be solved to optimality.

This chapter is organized as follows. Section 3.2 will introduce notation, and state the problem formally. Section 3.3 will state the problem as a mathematical program, characterize its complexity, and discuss the advantages and disadvantages of various heuristic approaches, including multiplier adjustment methods (MAM's), potentials methods, and the joint state space (JSS) search . Section 3.4 will focus on the JSS search for the uniform cost structure, and show that a proper formulation can result in significant dimensionality reduction. Sections 3.5 and 3.6 will first overview and then detail an iterative algorithm to optimally solve the multiple cost path consensus problem in the uniform cost case, briefly discussing algorithm runtime. Section 3.7 will discuss the exten-

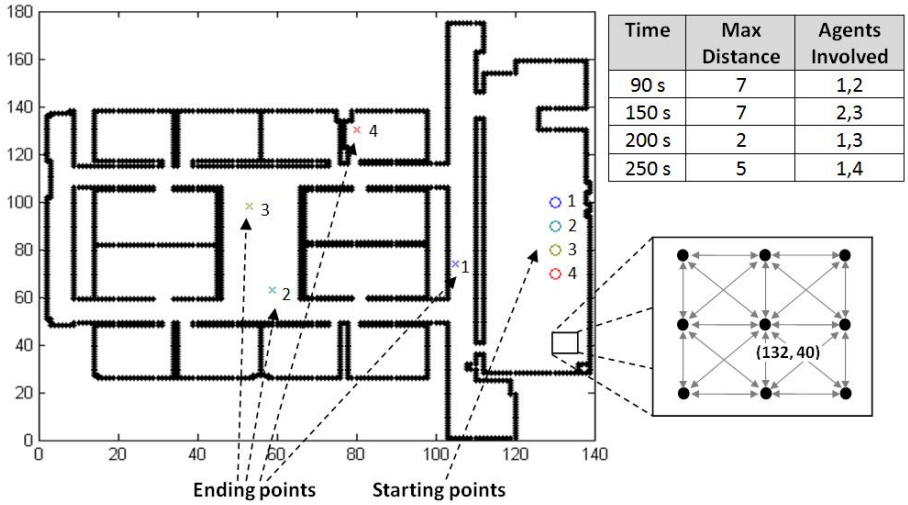


Figure 3.1: Example 180×140 gridworld simulating a complex office environment.

sion of the algorithm to the non-uniform cost case. Section 3.8 will verify the algorithm's performance through simulation, showing that for certain operating regimes easily attained in practice, the algorithm provides extremely rapid performance. Section 3.9 will conclude with a discussion of design choices to ensure good performance for live experiments and a survey of future work.

3.2 Problem Statement, Notation

We are given a graph $G(E, V)$ with E edges and V vertices around which N agents navigate. Though $G(E, V)$ may take any form, we will consider a gridworld - a quantized, two dimensional representation of a real-world environment, where vertices v can be indexed by their Cartesian coordinates (x, y). This also allows us to define an auxiliary distance metric $d(u, v)$ as the Cartesian distance between vertices u and v.

For each agent n we create binary decision variables f^n_{xyt} that determine

whether or not it is located at a particular vertex v (v is indexed by coordinates (x, y) at time t. Together, the set of N decision variables $\left\{f^1_{xyt^*}, ..., f^N_{xyt^*}\right\}$ form an N-Tuple $\mathcal{F}^N_{xyt^*}$, which we will use to represent the locations of the N agents at time t^*

We define an objective (total cost) function as weighed sum of all the decision variables. The weights correspond to an individual agent's cost of traversing an edge in G at time T.

The agents are linked by P time-parameterized constraints, and are required to navigate from $\{Start_i\} \in V$ to $\{Goal_i\} \in V$ at minimum total cost while satisfying these constraints. Any distance constraint at time t^* can be expressed by partitioning the space of N-tuples $\left\{\mathcal{F}^n_{xyt}\right\}$ into a set of 'allowed' tuples, and a set of 'prohibited' tuples. For example, if we require that two agents n and n' be no more D units apart at time t^*, the set of all decision variable pairs $\left\{f^n_{ut^*}, f^{n'}_{vt^*}\right\}$ such that $d(u, v) \leq D$ forms the allowed set, while its complement forms the prohibited set. We will use only these 'maximum-distance' types of constraints in this chapter. However, any other position-dependent constraints may be implemented using the prohibited tuples mechanic. An example 180×140 gridworld simulating an office environment, along with obstacles, N agents, their respective start and goal coordinates, and their linking constraints is shown in Figure 3.2. Integer coordinates (x, y) within office boundaries form the vertices of G. Each vertex is 8-connected to its nearest neighbors.

Finally, we make an important observation - for the MPC problem, the planning horizon is always finite. This is because an agent n's decision variables need not be augmented with the time dimension after the last constraint in which it is involved. Because it is no longer linked with the other agents, the last

leg of n's journey may be planned via a search shortest path search (which needs only the spatial dimensions) without compromising path optimality. Thus, each problem instance generates a finite number of decision variables, and thus, a finite combinatoric problem.

We can now define our objective in the following way: we are required to select a finite sequence of N-tuples such that path continuity is satisfied for all agents for all t, start-to-goal conditions are satisfied for all agents at some t, and all P time-indexed constraints are satisfied (the chosen N-tuples for all constraint times t_p fall into their respective allowed sets).

3.3 Possible Solution Approaches

3.3.1 Mathematical Programming

Given the above notation, as well as the MPC problem's similarity to the well-studied shortest path problem, it's natural to attempt casting the problem into a mathematical programming framework. Indeed, the sets of prohibited and allowed tuples may be expressed via linear constraints. A prohibited tuple $\mathcal{F}^n_{xyt}$ generates the constraint:

$$\sum_{n=1}^{N} \mathcal{F}^n_{xyt} \leq N - 1 \tag{3.1}$$

where the summation is carried out over the N members of $\mathcal{F}^n_{xyt}$. If there are only M agents with indices lying in the set $\mathcal{P}_p$ are involved in constraint p, the above expression can be rewritten as

$$\sum_{n=1}^{M} \mathcal{F}'^{n}_{xyt} \leq M - 1 \tag{3.2}$$

where $\mathcal{F}'^{n}_{xyt} = \left\{ f^{n}_{xyt}; f^{n}_{xyt} \in \mathcal{F}^{n}_{xyt}, n \in \mathcal{P}_p \right\}$. The dimensionality of the problem is thus reduced, since, each constraint now generates $O(|G|^M)$ linear subconstraints. We can write the MPC problem as a standard shortest path problem with additional linear constraints:

$$\min \sum c^{n}_{xyt} f^{n}_{xyt} \tag{3.3}$$

Subject to:

$$A f^{n}_{xyt} = b \tag{3.4a}$$

$$F f^{n}_{xyt} \leq g \tag{3.4b}$$

$$f^{n}_{xyt} \in \{0, 1\} \tag{3.4c}$$

Here, the equality constraints account for path continuity (conservation of flow), while the inequality constraints account for the MPC problem's linear subconstraints. This 'constrained shortest path problem' has been previously studied in optimization literature, and shown to be NP-complete ([21], p. 13). However, in this standard form, it is amenable to both standard subgradient descent algorithms [20] and more specialized multiplier adjustment methods (MAM's) [30] [16]. However, the authors of [30] and [16] assume a small number of coupling constraints - typically one to ten. Since any practically sized instance of our problem contains a potentially huge number of multipliers, and since each iteration of either subgradient descent for a given MAM requires the solution of N shortest path problems, these methods, though sound in theory, are impractically time-consuming in practice.

3.3.2 Potentials

The authors of [36] address the MPC problem directly, aiming to decouple the graph searches involved, while still guaranteeing eventual convergence to optimal paths under certain conditions. Their approach employs shortest path searches on N graphs whose edge costs are augmented by penalty functions, whose relative weight increases with each algorithm iteration. These penalty functions represent 'degree of failure' - how far a given agent's position at time t^* in its current path is from satisfying a maximum-distance constraint at t^* (i.e. how much farther than the allowed distance it is from the other agent(s) involved in the constraint). The agents perform searches on their respective augmented graphs in a round-robin fashion, with the penalty functions for each agent n adjusted at each search iteration. In effect, the algorithm generates increasingly powerful potential fields that affect the costs for all of n's decision variables generated at time t^*, forcing it closer to its constraint partner(s) if the maximum-distance constraint at t^* is violated. Their approach employs shortest path searches on N graphs whose edge costs are augmented by penalty functions, whose relative weight increases with each algorithm iteration. These penalty functions represent 'degree of failure' - how far a given agent's position at time t^* in its current path is from satisfying a maximum-distance constraint at t^* (i.e. how much farther than the allowed distance it is from the other agent(s) involved in the constraint). The agents perform searches on their respective augmented graphs in a round-robin fashion, with the penalty functions for each agent n adjusted at each search iteration. In effect, the algorithm generates increasingly powerful potential fields that affect the costs for all of n's decision variables generated at time t^*, forcing it closer to its constraint partner(s) if the maximum-distance constraint at t^* is violated.

The authors show the algorithm to converge in a matter of minutes for reasonably-sized instances (50×50 gridworld, 5 agents, 200 timestep planning horizon) . However, the algorithm has several important limitations. First, the stated theoretical condition for convergence to optimality is not easily translatable into practical terms. This means that there is no simple way to determine whether a particular problem instance may be solved optimally. Second, the potentials framework constrains the user to working only with maximum-distance type constraints.

3.3.3 Joint State Space Search

An alternative way to consider the problem is to search directly within the joint state space of decision variables, selecting a cost-optimal sequence of N-Tuples which generate paths satisfying continuity, start-to-goal, and maximum-distance constraints. At first glance, this approach seems completely intractable - for a problem instance with a 100×100 gridworld, 5 agents, and a planning horizon p' of 100 time units (which can alternatively be thought of as the last constraint in which any agent is involved), there are approximately $|G|^{Np'} = 100,00^{5 \cdot 100}$ N-Tuples to be evaluated. However, as we show in the following sections, a correct formulation, along with several assumptions about the problems structure, significantly reduce the problem's dimensionality and allow for a practical tree-search algorithm within the joint state space.

3.4 Joint State Space Planning

3.4.1 Identical Cost Search

We assume costs are identical for all actions within a given division of time Δt - an agent incurs the same cost for moving to any vertex reachable from its original position within Δt time units, or by standing still[2]. This cost structure is useful for setups where time is the dominant factor, or where the energy expenditure of an agent is roughly proportional to the time it operates. It is also useful in modeling setups with heterogeneous agents - agents that move at different speeds - without dealing with complicating factors, such as the need to quantize the time dimension according to the greatest common factor of agent speeds.

Previously, we've made the observation that agents do not need to augment their graphs with time after the last constraint that involves them, meaning that there is no need for limited planning horizons - each node has a finite graph on which to do its planning.

We exploit the structure of the problem to significantly reduce its complexity. First, we observe that the cost for an agent i to satisfy a constraint p at time t_p is exactly t_p , regardless of the path it takes. This is because the costs are measured in time, and regardless of its speed/path taken, agent i must be active at time t_p (and have therefore accumulated t_p units of cost) to satisfy the constraint. Therefore, the only uncertainty is how fast an agent can get to its goal after satisfying the last constraint in which it's involved (i.e. distance between goal

[2] An A^*-like algorithm for generalized costs may also be derived - see Section 3.7 for details.

and agent's position at time t^i_{last} - the last constraint involving i). The total cost is therefore $\sum_i t^i_{last} + dist\_from\_goal\_afterward_i$. This crucial observation allows the following reformulation of the problem:

Find an feasible N-tuple that minimizes $\sum_i dist\_from\_goal\_afterward_i$.

This formulation allows use of a type of bisection search coupled with a feasibility check. Since all path costs between adjacent time-slices are identical, we need only to generate a sequence of P N-tuples, tbe p^{th} N-tuple representing a configuration that's satisfactory with respect to inter-agent constraints and feasible with respect to the preceding and following N-tuple (i.e. the agents must be able to get from their positions at time $p-1$ to their positions at time p, and from these to their positions at time $p+1$. The intermediate positions may be filled in afterward, without loss of optimality.

Note: for an efficient implementation of the search, we will need to implement an all-to-all Dijkstra database for the original graph - it will have $|V|^2$ entries, which is large, but can be computed off-line.

3.5 Algorithm: Overview

3.5.1 N-Ball Search

Since each agent's non-time-augmented graph has $|V|$ vertices and there are N agents, there are $|V|^N$ possible positions for agents to be in at the time the last constraint is satisfied. Each of these configurations carries a cost

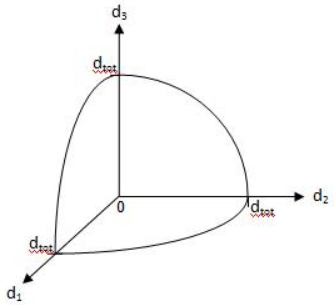


Figure 3.2: N-Dimensional ball of satisfying points

$\sum_i dist\_from\_goal\_afterward_i$.

If we require that a total cost be less than d_{tot}, we obtain an N-dimensional ball of satisfying points (or at least, the positive portion of that ball). These balls can be thought of as disks around the goal states - for a particular arrangement $(d_1, ..., d_N)$, $\sum_i d_i \leq d_{tot}$, the set of satisfying points can is contained in a series of disks of radii $d_1, ..., d_N$ around the goal states of the agents. Not all these points are satisfying - their straight-line path may be obstructed by an obstacle. The all-to-all Dijkstra database may be used to check for this. See Figure 3.5.1 for example of a satisfying set for a 3-tuple.

3.5.2 Search Mechanism

Assume we have a method for determining whether a given N-tuple is feasible, i.e., there exist n paths satisfying continuity, distance constraints which end at the N-tuple).

If a chosen n-tuple with cost D_{tot} is feasible, then it creates an upper bound on D_{tot}. We need not consider any n-tuple with a cost $\geq$ the current one.If a

chosen n-tuple with cost D_{tot} is infeasible then there may still exist another N-tuple with cost $\leq D_{tot}$ which is feasible. Current point's infeasibility may be due to the particular configuration, and not an overly restrictive D_{tot}. If all *all* the points on the interior of a ball of radius D_{tot} are infeasible, but a point on its surface is feasible, that point has optimum cost.

3.5.3 Checking Feasibility

We again use the structure of the problem to formulate a feasibility problem of lower complexity. As noted previously, we only need to examine $|\{t_t\}|$ slices of spacetime during our feasibility check. The search is conducted in reverse - from the goal (or rather, the N-tuple chosen by the N-ball search) to the start. We conduct a P-layer depth-first search, since the goal is to find a feasible path, and all paths between time slices Δt units apart have an identical cost equal to Δt. N traces - space-time points which the agents must visit - are maintained. For each constraint, we select an M-dimensional point in the joint state-space of the agents involved in the constraint, append this point to their traces, and conduct a continuity and constraint satisfaction check. When an agent's trace successfully reaches its starting point, it becomes inactive. The search proceeds until all traces are inactive, which signifies success, or until all possibilities are exhausted, which signifies failure.

3.6 Algorithm: Details

The algorithm consists of two functions: **N_BALL_SEARCH**, which accepts a graph G, a set of N agents and start-goal points, and a set of P inter-agent constraints, and returns either a set of N constraint-satisfying, minimum-length shortest paths, or the empty set if the problem is infeasible. The recursive function **FEASIBILITY_CHECK** is used within **N_BALL_SEARCH** to determine whether a particular set of initial positions $\mathbf{X}_{input} = (X_1, X_2, ..., X_N), X_i \in V$ is feasible (where feasibility is defined in the previous section).

Before giving the algorithm, we define several relevant terms. First, the P inter-agent constraints are given by three arrays: $T\_CONSTR = [t_1, ..., t_P]$ - the constraint times, $I\_CONSTR = [I_1, ..., I_P]$ - the indices of the of the agents involved, and finally $S\,(T\_CONSTR(p), \mathbf{X})$ - a binary-valued function that determines whether a configuration $\mathbf{X}$ is feasible under constraint p. For $S\,(\cdot)$, $\mathbf{X}$ the form $(v_1, ..., v_M)$, where $v_i \in V$ and M is the number of agents involved in constraint i. We next define the function $Feas\,(v, \Delta t)$ as the set of vertices reachable from v in Δt units of time. Additionally, we define the sets $\left[t^1_{first}, ..., t^N_{first}\right]$ and $\left[t^1_{last}, ..., t^N_{last}\right]$ where t^i_{first} is the first constraint, time-wise, in which agent i is involved, and t^i_{last} is the last one.

PATHS is an N-dimensional stack, which keeps track of constraint points (points which must be visited) assigned to all agents, as well as associated constraint times (the times when the agents must be at these points). Two standard operations, PUSH(**PATHS**, I,**X**,**T**) and POP(**PATHS**, I,**X**,**T**) are provided with the stack. If **X** is of dimension less than N, the accompanying indices I are used to push the position and time points onto the correct part of the stack. Finally,

FILL_PATHS is a subroutine which converts the stack **PATHS** into a collection of N complete, time-indexed paths. FILL_PATHS operates by sequentially working though each agent's stack, computing shortest paths from the agent's goal to each of its constraint points, and finally to its start. The problem of having the agent get to its constraint point early, Δt units before the constraint time, is handled by adding Δt units of delay to the shortest path.

Function:N_BALL_SEARCH(·)

ACTIVE = $1, ..., N$
$STATE\_SPACE = G_1 \times ... \times G_N$
PATHS = $\{\emptyset\}$
T = $\left[t^1_{last}, ..., t^N_{last}\right]$
Choose maximum cost D_{tot}
while STATE_SPACE $\neq \emptyset$ **do**
    Choose **X** $\in$ **STATE_SPACE** s.t. $\sum_i d\left(\mathbf{X}_i, Goal_i\right) \leq D_{tot}$
    PUSH(**PATHS**, N, **X**,**T**)
    ProbFeasible = FEASIBILITY_CHECK(**X**,**T**, P)
    if ProbFeasible = 1 **then**
        Remove all **X'** s.t. $\sum_i d\left(\mathbf{X'}_i, Goal_i\right) \geq D_{tot}$ from **STATE_SPACE**
        Choose new $D_{tot} \leq$ previous D_{tot}
    else
        Remove **X** from **STATE_SPACE**
        POP(**PATHS**, N, **X**,**T**)
return FILL_PATHS(**PATHS**)

Function:FEASIBILITY_CHECK(X,T, ACTIVE, p)

1: $t_p = T\_CONSTR(p)$
2: $I_p = I\_CONSTR(p)$
3: **if** $S\left(t_p, \mathbf{X}\left(I_p\right)\right) = 0$ **then**
4: **return** 0
5: **else**
6: **for** each $i \in$ **ACTIVE′ do**
7: **if** $t_p = t^i_{first}$ **then**
8: **if** $Start_i \in Feas\left(X_i, t_p\right)$ **then**
9: Remove i from **ACTIVE**
10: **else**
11: **return** 0
12: **if ACTIVE** $= \emptyset$ **then**
13: **return** 1
14: $j = 1$
15: **for** each $i \in I_{p-1}$ **do**
16: $\Delta t^p_i = t_{p-1} - T_i$
17: $FeasTuples_j = Feas\left(\mathbf{X}_i, \Delta t^p_i\right)$
18: $j = j + 1$
19: **for** each $\mathbf{X''} \in FeasTuples$ **do**
20: $\mathbf{X'} = \mathbf{X}$
21: $\mathbf{T'} = \mathbf{T}$
22: $\mathbf{X'}\left(I_{p-1}\right) = \mathbf{X''}$
23: $T'\left(I_{p-1}\right) = t_{p-1}$
24: PUSH(**PATHS**, I_{p-1}, $\mathbf{X''}$,$T'\left(I_{p-1}\right)$)
25: $Value$ =**FEASIBILITY_CHECK**($\mathbf{X'}$,$\mathbf{T'}$, **ACTIVE**, $p-1$)

26: **if** $Value = 0$ **then**

27: POP(**PATHS**, I_{p-1}, **X**'',$T'\left(I_{p-1}\right)$)

28: Continue

29: **else**

30: **return** 1

31: **return** 0

We note several points of interest. First, the exact methods for cycling through the candidate N-tuples in Line 7 of **N_BALL_SEARCH** or candidate M-tuples in Line 19 of **FEASIBILITY_CHECK** are left to the algorithm designer. We suggest several helpful selection heuristics in the next section. Second, within **FEASIBILITY_CHECK**, instead of selecting points in the joint state space of dimension $M = \left|I_p\right|$ and proceeding via a single recursion on p, we may proceed via a double recursion on p and $i \in I_{p-1}$ (Line 15). We first recurse on i, cycling through all $i \in I_{p-1}$ and checking whether each individual position m_i is feasible with respect to the ones already chosen. Once a satisfactory M-tuple is built, we recurse on p, proceeding to $p - 1$. Each recursion level is labeled by the pair (p, i) This approach is functionally equivalent to single recursion, while making it conceptually easier to implement various selection and pruning heuristics. In the next section, we assume that a double recursion search is implemented.

3.6.1 Implementation Details

For exhaustive search algorithms such as the one presented in this chapter, the right search order and pruning heuristics can make a significant difference in

performance, determining how large a problem instance can be solved in reasonable time. Here, we present several heuristics based on geometric intuition about the problem which we feel improve performance significantly. These are:

Initial Point Picking

Line 7 of **N_BALL_SEARCH** dictates that one choose N-tuples within the joint state space. Likewise, Line 19 of **FEASIBILITY_CHECK** dictates that one cycle through the set of M-tuples for agents involved in constraint p. However, the question of where to begin the search is left open. We propose that in both cases, one uses geometric intuition to determine a likely rendezvous point. This point should take into account both the current and start positions (since the search proceeds from goal to start) of the agents involved, as well as the time each of them has available.

We propose an iterative point-picking algorithm, whereby an initial point u is computed as the convex combination of the agents' current and starting positions, with the current positions receiving slightly greater weight. If all agents may reach a vertex within a disk of radius d^p_{max} (the maximum distance allowed by the current constraint) around u, the algorithm returns u. Otherwise, the relative weights given to the current positions of the agents unable to reach the disk are increased, and the convex combination is recomputed. The algorithm proceeds until a satisfactory point is found, or until a predetermined number of iterations is reached (in which case, the last computed u is returned).

This algorithm may be used in **FEASIBILITY_CHECK** during recursion on $i \in I_{p-1}$ (Line 15). We first determine the position u of the first agent selected from

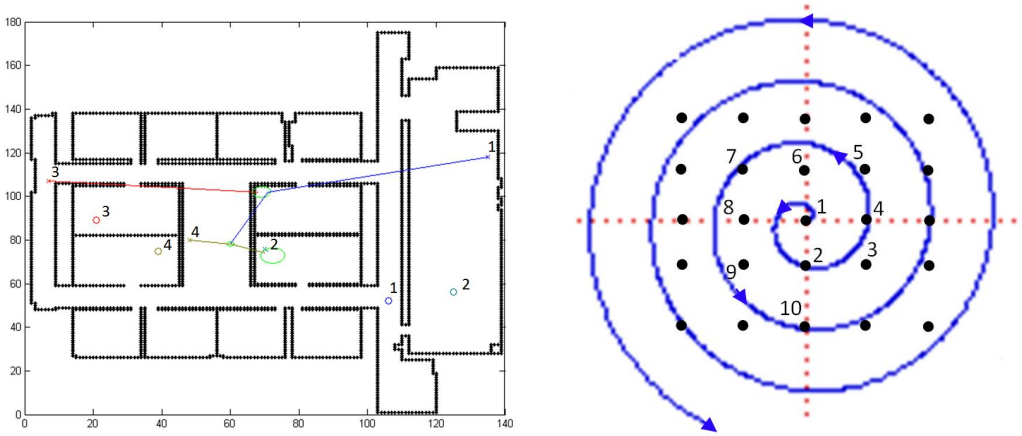
I_{p-1}. Each subsequent $i \in I_{p-1}$ starts at u, and has its reachable set of positions (Line 17) intersected with a disk of radius d^p_{max} centered at point u.

The algorithm may also be applied repeatedly to obtain the first feasible N-tuple in **N_BALL_SEARCH**. We initialize an empty N-tuple $\mathbf{X}_{init} = -1$ and two sets of positions - $\mathbf{X}_{curr} = \{Goal_i\}$ and $\mathbf{X}_{start} = \{Start_i\}$. Starting at constraint P, we apply the point picking algorithm, computing u_P as a convex combination of $\mathbf{X}_{curr}$ and $\mathbf{X}_{start}$. We then assign u_P as to any agent $i \in I_P$ s.t. $\mathbf{X}_{init}(i) = -1$, and $\mathbf{X}_{curr}(I_P) = u_P \forall i \in I_P$. We then proceed to constraint $P - 1$, repeating the procedure until all agents are assigned an initial position. Example output of this algorithm is shown in Figure 3.3a.

Selection Order Heuristic

The initial point picking heuristic provides an initial candidate position. In the case of **FEASIBILITY_CHECK**, the first agent whose position is thus fixed defines a disk of radius d^p_{max} within which the positions of all the other agents involved in constraint p must be contained. However, the question of how to choose subsequent candidate positions, should the initial one prove infeasible, is again left open.

Based on simulation experience, we argue that the initial point picking heuristic provides 'almost feasible' positions - positions that may be reachable in terms of distance, but lie within an obstacle. If this is the case, moving some small distance from the initial selection may lead to a satisfactory solution. We propose that candidate points are selected according to their position on an outward spiral emanating from the initial point, as shown in Figure 3.3b.



(a) (b)

Figure 3.3: Point selection heuristics: (a) shows the output of the initial point picking heuristic; (b) shows the candidate point selection order.

Pruning Heuristic

Currently, if the double recursion in **FEASIBILITY_CHECK** hits a 'roadblock' at level $(p-1, i)$, i.e. if no feasible position $m_{I_{p-1}(i)}$ exists for agent $I_{p-1}(i)$ within the disk of radius d_{max}^{p-1} centered on $m_{I_{p-1}(1)}$, it recurses back to to level $(p-1, i-1)$ (or to $\left(p, \left|I_p\right|\right)$ if $i = 1$) and chooses another point. This is not efficient, since the previous recursion level may not involve either agent $I_{p-1}(1)$ or agent $I_{p-1}(i)$, and no points chosen at that level will not impact either $m_{I_{p-1}(1)}$ or the reachable set of agent $I_{p-1}(i)$ at level $(p-1, i)$. The algorithm may thus spend a large amount of time looping between $(p-1, i)$ and previous levels, iterating over points which have nothing to do with the current roadblock.

The following pruning heuristic remedies this problem: if no feasible position $m_{I_{p-1}(i)}$ for $I_{p-1}(i)$ exists within d_{max}^{p-1} units of $m_{I_{p-1}(1)}$, backtrack to level (p', i') such that either $i' = I_{p-1}(1)$ or $i' = I_{p-1}(i)$. Depending on the particular problem instance, this pruning heuristic may trim away large portions of the search tree without sacrificing either the algorithm's completeness or its optimality.

3.6.2 Running time

The running time of the feasibility check depends on the number of agents involved in each constraint and the number of constraints. Let $M_1, ..., M_P$ be the number of agents be involved in constraints 1 through P. At each time slice p, the algorithm considers $O\left(|G|^{M_p}\right)$ points. Therefore, $O\left(|G|^{\sum_p M_p}\right)$ points are considered.

Assume, as in the example, that we limit this number two (though there is no reason why it can't be more). For an exhaustive search, we need to try at most $|V|^2$ points at layer one. For each of these, we need to try at most $|V|^2$ points at layer two, etc. Accordingly the running time is proportional to $\left(|V|^2\right)^p$, where p is the number of inter-agent constraints. Thus, the running time is polynomial in the number of grid points, exponential in the number of inter-agent constraints, and largely unaffected by the number of agents.

The N-Ball search, on the other hand goes through at most $|G|^N$ nodes, meaning that it's exponential in the number of agents. However, in practice this figure can be significantly reduced through proper choice of D_{tot} at each iteration of the search, or eliminated entirely (albeit while sacrificing the optimality guarantee) through use of heuristic algorithms such as the one described in Section 3.6.1.

The following equation gives running time for our algorithm:

$$O\left(|G|^{N \sum_p M_p}\right) \tag{3.5}$$

3.7 Generalized Costs

In this section, we show how to extend the solution framework to a graph with a non-uniform edge costs. Though in general the edge costs can take any form ($c(e) \in R$), we will assume that they are some function of the distance a agent must move combined with the time a movement takes: $c(e) = f(c_{distance}, c_{time})$

In the following, we show that this problem can be solved via a A^*-like search.

3.7.1 Constructing a Generalized Cost Graph

In Section 3.4.1, it was shown that time need not be addressed explicitly in the state space. Rather, we create and examine P time-slices (each corresponding to an inter-agent constraint), each with of $|G|^M$ verticies, where M is the number of agents involved in each constraint. Time figures into the problem indirectly - we use the time difference Δt between any two slices to determine the set of points reachable by the agents from one one slice to the next (basically, the range of the agent's travel within the time limit Δt given their position in the previous slice). The same exact technique can be used for generalized edge costs,the only difference being that the set of reachable points will now have different costs depending on path taken.

We proceed to explicitly construct a graph on which the search will be undertaken. We begin at the goal states. Let $\mathbf{T}_i$ be the times of constraints involving agent i, starting with t_{last} and beginning with t_{first}. For each goal state $Goal_i \in V$, create a set $X_i(t_{last}) = V$, connecting $Goal_i$ to every edge in $X_i(t_{last})$,

with $c(Goal_i, X_i(t_{last}))$ being equal to the shortest distance in G between them. For each $X_i(t_{last})$, we generate a set $X_i(\mathbf{T}_i(2)) = Feas(X_i, (\mathbf{T}_i(1) - \mathbf{T}_i(2)))$, with each $v \in X_i(t_{last})$ connected to the verticies $Feas(v, (\mathbf{T}_i(1) - \mathbf{T}_i(2)))$ in $X_i(\mathbf{T}_i(2))$, with by edges costs equal to the shortest paths the two. We proceed in this way, until we reach t_{first} for all i. The result is a set of N graphs, which we will collectively term G'.

3.7.2 Generalizing A^* Search

Like the identical-cost algorithm given in Section the generalized-cost search constructs the agents' paths nonuniformally - it progressively defines where the relevant agents need to be during a particular constraint time.

Each N-tuple $\mathbf{X}$ now has associated with it a two part cost $h(\mathbf{X})$ - the total distance/time cost already travelled by the N agents, and a heuristic estimate of the total distance/time cost left to travel. For each $v \in X_i(\mathbf{T}_i(p))$, we define this heuristic as the shortest distance/time path from v to $Goal_i$, plus the difference between $\mathbf{T}_i(p)$ and the time cost of of the shortest distance/time path. As with the A^* algorithm [41], the heuristic is always a lower bound on the actual time/distance cost left to travel.

We are now ready to define the search procedure. We begin with $\mathbf{X} = \{Goal_i\}$. At each constraint layer p, generate the set of an M-tuples $\mathbf{X}''$ satisfying the constraint, and map them to a set of N-tuples $\mathbf{X}'$ such that $\mathbf{X}'(I_p) = \mathbf{X}'', \mathbf{X}'(I_p^C) = \mathbf{X}$. We then select the N-tuple which has the lowest cost $h(\mathbf{X}')$ among all N-tuples considered so far at the current or higher layers. The search proceeds to the lowest layer (the very first constraint). At this time, if a satisfying solution exists

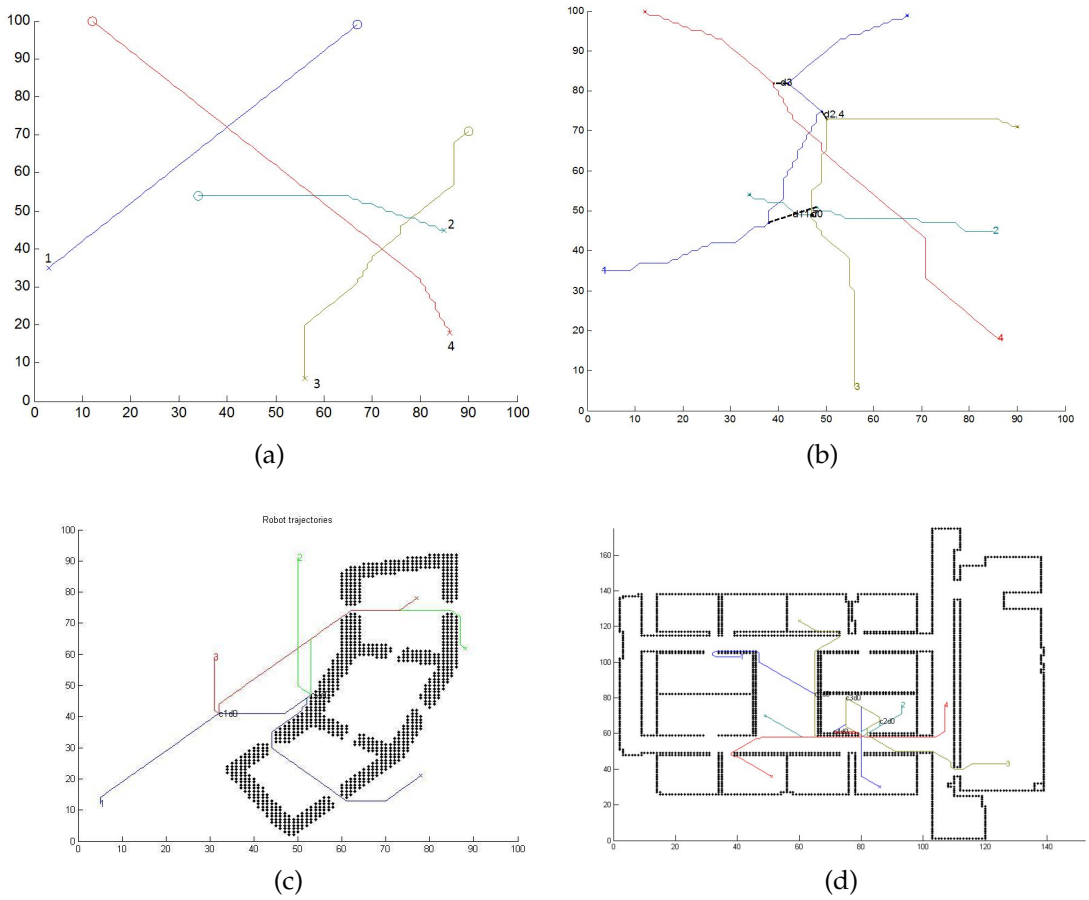


Figure 3.4: Simulation Results: (a) shows unconstrained agents navigating in free-space; (b), (c), and (d) show constrained multi-agent navigation in free-space, small office environment, and large office environment, respectively.

and satisfies the minimality criteria, it is guaranteed to be the minimum solution to the problem.

3.8 Simulation Results

To determine the performance and robustness of our algorithms, we tested it with problem instances and environments of varying size and complexity. In this chapter, we provide three representative environments: Free-space, a small

office environment, and a large, complex office environment.

Note: Due to time constraints, we were able to run only the **FEASIBILITY_CHECK** portion of our algorithm, meaning that we obtained feasible solutions as opposed to optimal ones. However, our path lengths and compared favorably to those in [36], and a good heuristic for **N_BALL_SEARCH** may easily be implemented as described in Section 3.6.1.

Fig. 3.4a shows unconstrained movement of four agents in 100×100 node free-space between their respective s-d pairs. Fig. 3.4b shows the same agents, now coupled by constraints. Because the heuristics in Section 3.6.1 are tailored toward free space, our algorithm performed extremely efficiently in instances like this. Fig. 3.4c shows three agents navigating in a 100×100 node office environment of medium complexity. This environment is identical to the one given in [36], and is useful for performance comparisons. Fig. 3.4d shows four agents, bound by the constraints given in 3.2, navigating in a more complex 180×140 office environment.

We first tested the effect that inter-constraint timing (i.e. the tightness of the constraints) has on the algorithm's performance and chances of success. We began with the office environment and inter-agent constraints given in Fig. 3.2. We then multiplied each constraint time by a parameter α (leading to correspondingly shorter or longer inter-constraint times), and ran our algorithm for several values of α, averaging over 50 instances generated by choosing random s-d pairs for each of the four agents. Fig. 3.8 shows the results of these runs. Note that as α approaches zero, the average probability of successfully solving a given instance tends toward zero, while both the time and number of necessary recursive function calls increases exponentially. Converseley,

larger inter-constraint times yield near perfect solvabilty while requiring orders of magnitude less calculation. The graphs provide a useful guideline for designing/adjusting problem instances - if a given instance is difficult/time-consuming to solve, expanding the time budget by as little as 10% may yield significant performance/solvability improvements.

Next, we surveyed the algorithm's performance across different environments and for differing numbers of inter-agent constraints. We tested the environments shown in Figs. 3.4b, 3.4c, and 3.4d. For each, we tested three different constraint sets. In the 'easy' set, agents 1 & 2, 3 & 4, and 1 & 4 are required to approach each other to within 5 units at 50 sec. intervals, starting at 50 sec. from start. In the 'moderate' set, agents 1 & 3 are additionally required to approach each other at 200 sec. from start. In the 'difficult' set, agents 2 & 4 and 3 & 4 must additionally approach each other at 250 and 300 sec., respectively. For each of these setups, we averaged over 50 random s-d pair instances. Table 3.1 shows the results of the simulation, giving both the average probability of success and the average number of recursive iterations required. Note that simple free-space is able to handle most instances with ease, due to the selection/focusing heuristics employed, while more difficult environments require significantly more recursive iterations for convergence. Also, note the highly exponential increase in the amount of computation necessary as the number of constraints increases.

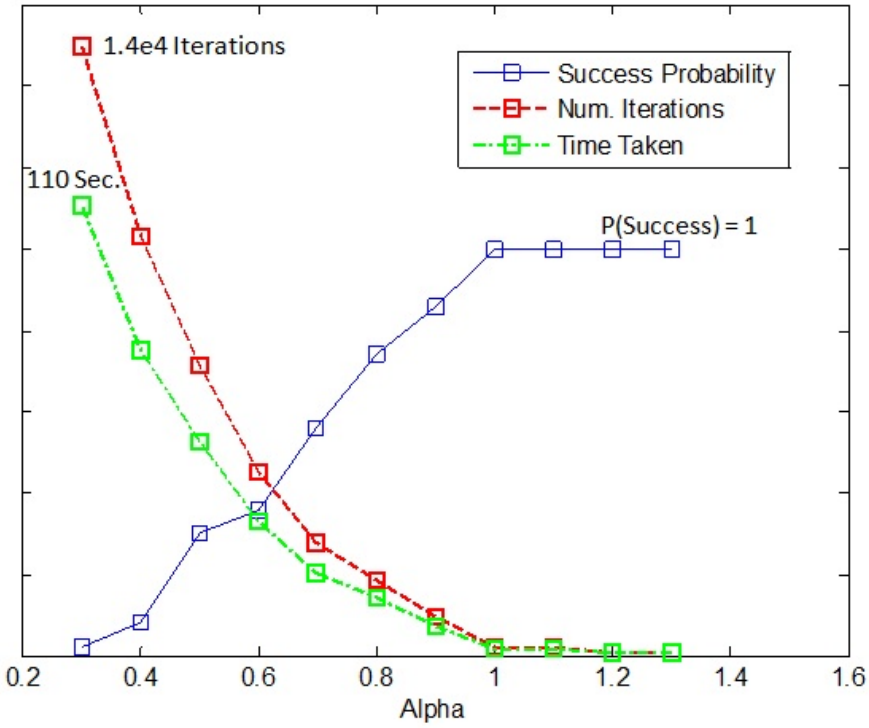


Figure 3.5: Algorithm performance as a function of inter-constraint timing, Alpha. The environment, number of agents, and number of constraints are kept identical for all values of Alpha

Table 3.1: Algorithm performance for various environments and various numbers of constraints

Num. Constr.:	**3**	**4**	**6**
Free Space	1 / 6.6e1	1 / 8.0e1	1 / 3.9e2
Small Office	.98 / 5.2e3	.94 / 3.9e4	.76 / 5.8e
Large Office	.90 / 3.7e4	.78 / 1.4e5	.73 / 8.1e7

3.9 Discussion and Conclusions

The experiments in the above section have shown that the JSS search we've developed, along with the geometric point-picking heuristics described in Section 3.6.1 yield excellent results for free space or lightly cluttered environments. However, for more complex environments, certain design choices and algorithm modifications may yield vast increases in performance.

For instance, if the designer chooses inter-agent constraints which are reasonably spaced time-wise, they may avoid a critical region [CITE] problem instance, which has few difficult to find feasible solutions. Additionally, the designer may choose to 'relax' constraints, either increasing the maximum allowed distance or increasing the inter-constraint time budget given to a set of agents if a particular constraint proves troublesome (i.e. said agents are unable to satisfy the constraint after some number of attempts).

Additionally, possible future work may include implementing minimum-distance constraints as mentioned in Section 3.1 (the search algorithm will remain exactly the same, but slightly different point-picking/focusing heuristics will be required). It may also include implementing the non-uniform cost via an A^*-like algorithm, and developing heuristic methods for mitigating its exponential nature.

CHAPTER 4

THE HETEROGENEOUS SENSOR, HETEROGENEOUS SENSING REGION PROBLEM

4.1 Introduction

The work in this chapter addresses reconfiguration of an MWSN in a heterogeneous FoI so as to achieve optimum coverage at minimum cost while guaranteeing connectivity. By 'reconfiguration', we mean the physical relocation of node platforms, done in an autonomous way, with the purpose of improving a given metric. By 'heterogeneous' we mean an FoI partitioned into distinct sub-regions (sensing regions), each of which has a distinct set of sensing requirements, as in Fig. 4.5a. This situation can model applications that require, for example, one sensing region monitoring temperature measurements, while another sensing for temperature, light, and vibration. For ease of reference, we call such an FoI a Multiple Sensing Region FoI (MSRF). Consider a network of N mobile sensors randomly distributed across the FoI. Each sensor has a distinct set of sensing capabilities. The general problem can then be formulated as follows:

> *Given a set of sub-regions, each with a vector of sensing requirements R_j, and a distribution of N sensors, each with a vector of sensing capabilities S_i. What is the spatial arrangement which yields optimum coverage and satisfactory connectivity at minimum cost.*

*Unless otherwise specified, the work in this chapter was done in collaboration with Sergio Bermudez, who is a doctoral student at the Cornell School of Electrical and Computer Engineering, and Stephen Wicker, who is a professor at the Cornell School of Electrical and Computer Engineering. Additional collaborators will be acknoweledge at the start of specific sections.

Here, the definition of cost is left purposefully vague, since there are several meaningful ways to define it. For instance, in an energy-constrained network the dominant concern of the designer is minimizing total power consumption, so the cost function can be formulated as a weighed sum of the Euclidian distances d_i between each node's initial and final positions. However, for a solar powered-network with limited mobility [14], time to steady state (the time until a threshold fraction of the nodes converges to their final positions) might be the dominant concern. In this case, a function of the form $\max(d_i)$ is more appropriate. We consider both of these cost structures in Section 4.3.3.

Although to our knowledge the MSRF problem has not been discussed in literature, there exists a body of work dealing with related problems. Until now, network reconfiguration has been studied mostly in robotics literature, within the context of assigning tasks in multi-robot systems. For example, [22] solves the task allocation problem using a linear programming framework (for sensor networks, a set of physical positions inside the FoI can be thought of as 'tasks'). Another approach related to our problem is network deployment through the use of potentials [26, 42]. This method achieves some degree of optimality in coverage while maintaining connectivity, but has the disadvantages of being a heuristic method (no guarantees on the optimality of solutions) and only being able to handle a homogeneous FoI (all regions have the same requirements)

These works never address the generation of the set of physical positions. Also, the algorithms arrive at optimal assignments through global communication, which is infeasible or impractical for large numbers of low-powered nodes.

We propose to solve the problem in several stages. In Section 4.2, we offer a construction of the MSRF which facilitates both analysis and implementation.

Then, in Section 4.3, we show that under certain conditions the general problem may be decomposed into the spatial and assignment sub-problems. We give a solution to the former by adopting a methodology proposed in [11] and formulate the latter as either a linear or convex integer program (IP), depending on the form of the cost function. In the convex case we develop a distributed implementation suitable for a decentralized network. Simulation results are shown in Section 4.4. We conclude with discussion and future work in Section 4.5.

4.2 Defining the MSRF with Beacons

An important aspect of the problem setup is defining the method by which each node perceives the FoI and orients itself in it. We propose an MSRF setup which facilitates both analysis and implementation. Suppose there is a need for B sensing regions. Then we use a collection of B beacons, which may be manually placed devices or high capability mobile nodes. We define the sensing regions SR_b to be the Voronoi cells generated by the collection of beacon locations $\{b\}$:[1]

$$SR_b = \{x : x \in FoI,\ |x - b| \leq |x - b'|,\ b' \neq b\} \tag{4.1}$$

Each beacon transmits a pilot signal containing its absolute position and a set of requirements for the region it generates. The one disadvantage of this approach is that sensing regions are restricted to be simple polygonal shapes. However, the need for arbitrarily shaped sensing regions is not apparent and one can get an arbitrarily good approximation of any shape by using multiple beacons to generate regions with identical sensing requirements. There are,

[1] In order to avoid unbounded cells, we use a centrally located beacon to define the boundaries of an FoI, which we assume to be circular. Other shapes may be achieved through more complicated schemes.

however, multiple advantages. Region boundary definitions are simple, so a node at position x_i simply needs to compute the bisector of the segment $\left[b_{j_1}, b_{j_2}\right]$ to determine region boundaries and $\min \left|x_i - b_j\right|$ to determine region belonging. Also, beacons provide a simple mechanism for node localization as well as for placement of the optimal position lattice. Each node can use the beacons as anchors to compute its position in the FoI, e.g. using angle-of-arrival and time-of-arrival measurements.

4.3 Problem Decomposition

4.3.1 Assumptions

Before we begin a detailed analysis, we would like to state our assumptions.

1. The FoI is clear from obstacles which may obstruct movement or communication.
2. Each node can accurately determine both its own position and the position of every beacon.
3. Node density is sufficiently large such that an assignment satisfying all sensing requirements of all sub-regions can be made.[2]
4. There exists an appropriate infrastructure which allows efficient multi-hop communication, data collection and processing, etc.

Though this chapter considers only questions related to deployment, we believe that our connectivity guarantees allow for implementation of a variety of

[2]Later we drop this assumption, see Section 4.5.

Network and Application Layer algorithms.

4.3.2 Decomposition

Once every node has determined its own position and built a map of the MSRF, it needs to select an appropriate location to move to. For a single requirement-sensing modality, it has been shown in [11] that the arrangement of positions providing optimum coverage and guaranteed connectivity is a hexagonal grid, shown in Fig. 4.2. The inter-node spacing constants α and β in this hexagonal grid depend on both the sensing radius r_s and communication radius r_c (which we assume to be sufficiently large for the network to be fully connected). Thus, when each sensing modality is considered separately, a hexagonal grid with some value of α and β is optimal.

In order to make our problem tractable, we make a further assumption that all nodes have a common value of r_c and all sensing modalities have the same r_s. Using this assumption we can conjecture that, up to edge effects, the same position grid can achieve optimality for all sensing modalities. This fact allows us to *decompose* the problem into the Spatial and Assignment sub-problems. The spatial-sub problem is solved by generating a hexagonal 'optimal position grid' in the FoI, while the assignment sub-problem seeks to assign sensor nodes to every point on this optimal grid such that all sensing requirements are satisfied at minimal cost. A flowchart illustrating the decomposition is given in Fig. 4.3.2.

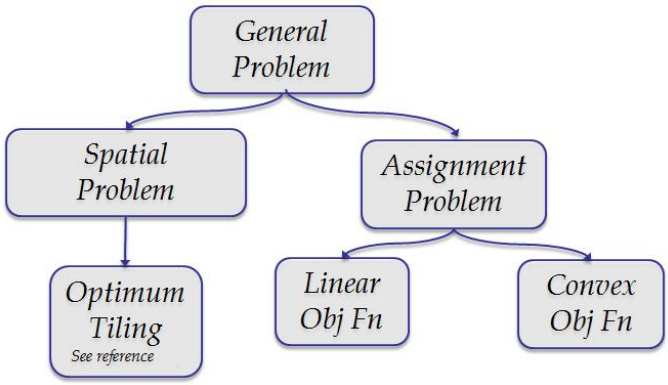


Figure 4.1: A flowchart showing the decomposition of the MSRF problem into Spatial and Assignment subproblems.

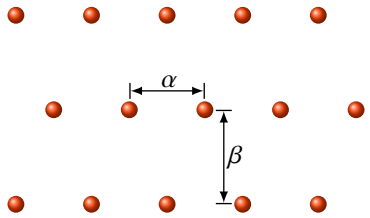


Figure 4.2: Hexagonal grid: optimum coverage, 2-connectivity.

4.3.3 Mathematical programming

To solve the Assignment problem, we first propagate the regions' sensing requirements to each grid position (see Fig. 4.5b for visual reference). Note that if a disk of radius r_s around a grid position intersects more than one sensing region, we require the sensor(s) placed at that location to satisfy all the regions' requirements. Also note that because of our Voronoi region formulation, each node can individually construct a globally consistent version of the optimal position grid, including the requirements of each position, using simple distance-to-beacon calculations.

Next, we introduce notation that will facilitate a rigorous formulation. We take a network with N sensor nodes, M optimal positions, and K possible sensing requirements. We define f to be the $N \times M$ binary Assignment matrix ($f_{ij} = 1$

means that sensor i is assigned to position j); R to be the $M \times K$ binary Sensing Requirements matrix ($R_{jk} = 1$ denotes position j having requirement k); S to be the $N \times K$ binary Sensing Capability matrix, and C to be the $N \times M$ Cost matrix (sensor i pays cost C_{ij} to move to location j). Note that C_{ij} is in all cases defined to be the Euclidean distance between the node's initial position i and optimal position j. These notation shorthands are summarized in Table 4.1 for easy reference.

We require that all sensing requirements be satisfied for every optimal position. More than one sensor can share the same location, but each sensor may be assigned to only one position.

To complete the mathematical programming (MP) formulation, we need to define an objective function. We define the objective function as follows.

Table 4.1: Notation

Symbol	**Description**
N	number of mobile sensors
M	number of 'optimal' positions within the FoI
K	number of requirements
i	sensor index
j	optimal position index
k	requirement index
f	$N \times M$ Assignment matrix
R	$M \times K$ Sensing Requirements matrix
S	$N \times K$ Sensing Capability matrix
C	$N \times M$ Cost matrix

Linear objective function

Consider a network of nodes with finite energy reserves. Then the cost function is defined

Here we assume that the energy a node has to spend is directly proportional to the distance of its travel, and that it is the same for all nodes in the network.

In this situation, the cost is defined as the total distance of travel by all the nodes. The distance of travel of each node is measured from its initial deployment to the final destination. The purpose of defining the distance of travel as a cost is to minimize the total energy expenditure of the network. Here we assume that the energy a node has to spend is directly proportional to the distance of its travel, and that it is the same for all nodes in the network.

To address this optimization, we define a linear program with the following parameters:

- the objective function minimizes the sum of the distances traveled by all the nodes
- constrained to the satisfaction of the coverage requirements of the FoI
- such that, each node is at most in a unique position of the grid
- and more than one node is allowed in any given position of the grid

This is a 0-1 programming problem and can be solved in a centralized way.

Convex objective function

In this case, the cost is considered as the time until the whole network reaches steady state. This is of relevance when the nodes are not constrained on energy, i.e. if there is a energy-harvesting mechanism. We consider that all the nodes in the network travel at the same speed and there are no collisions.

To address this optimization, we define a convex programming problem with the following parameters:

- the objective function minimizes the time until the system reaches steady state
- constrained to the satisfaction of the coverage requirements of the FoI
- such that, each node is at most in a unique position of the grid
- and more than one node is allowed in any given position of the grid

As in the previous case, this convex problem can be solved in a centralized way.

4.3.4 Mathematical programming formulation

We can write the resulting linear and convex integer problems as:

$$\begin{array}{ll} \textit{Linear objective function} & \textit{Convex objective function} \\ \text{(a)} \min \sum_i \sum_j C_{ij} f_{ij} & \text{(b)} \min \max C_{ij} f_{ij} \; \forall i, j \end{array} \tag{4.2}$$

Both subject to:

$$\sum S_{ik} f_{ij} \geq R_{jk}, \ \forall j, k \tag{4.3a}$$

$$\sum_{j} f_{ij} \leq 1, \ i = 1, \ldots, n \tag{4.3b}$$

$$f_{ij} \geq 0, \ f_{ij} \in \mathbb{Z} \tag{4.3c}$$

4.3.5 Problem Complexity

Before we proceed further, we characterize the complexity of the MSRF problem. As we will show below, the problem is NP-Complete, and thus warrants the development of heuristic algorithms for its solution.

We begin with several definitions. The Satisfiability pormblem (SAT) is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to 1. We will be dealing with formulas in Conjunctive Normal Form (CNF), which consist of possibly complemented literals (variables to be assigned), which are grouped into OR clauses, which are then joined by the AND operator. A sample CNF formula is given below:

$$(x_1 \vee x_2 \vee \neg x_3 \vee) \wedge (\neg x_2 \vee x_5) \wedge (x_3 \vee \neg x_5 \vee \neg x_6) \tag{4.4}$$

The CNF-SAT Problem is known to be NP-Complete. We will show that the decision version of the MSRF problem - determining whether there exists an

**The work in this section was done in collaboration with Richard Karp, who is a professor at the Berkeley Department of Electrical Engineering and Computer Sciences

assignment of sensors which satisfies all requirements - is also NP-Complete, making the optimization problem NP-Hard.

Theorem: The MSRF problem is NP-Complete

This theorem will be proved via the following claims:

Claim 1: MSRF $\in$ NP.

Proof: Given a certificate f, one may check that it satisfies all constraints in (ref) in polynomial time, since constraints are linear combinations of the entries in f.

Claim 2: The CNF-SAT Problem reduces in polynomial time ($\leq_p$) to the MSRF Problem.

The proof will be given in two steps. First, define "All-or-None Satisfiability" as the special case of CNF-SAT in which the literals in every clause are either all uncomplemented or all complemented.

Lemma 1: CNF Satisfiability $\leq_p$ All-or-None Satisfiability

Proof: replace every complemented literal $\neg x$ by a new positive literal x'. For each variable x add the following two clauses: $(x \vee x') \wedge (\neg x \vee \neg x')$. This ensures that the positive variables x and x' receive opposite values.

Lemma 2: All-or-none-Satisfiability $\leq_p$ MSRF problem in the case of two optimal positions

Proof: Define a separate resource r_c for each clause c. For each positive clause (i.e, no complemented variables), Position 1 requires one unit of r_c; for each

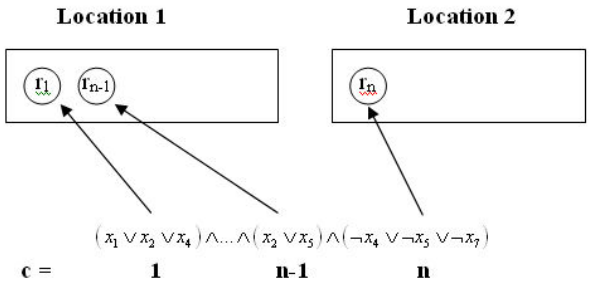


Figure 4.3: Reduction of the satisfiability problem to the MSRF problem.

negative clause c (i.e., all variables complemented) Position 2 requires one unit of r_c (see Figure 4.3.5). Define a sensor x for each variable x. For each clause c containing x or $\neg x$, x provides one unit of resource r_c.Except as defined above there are no additional requirements or resources.

Since the MSRF problem is NP-complete for two optimal positions, it is NP-complete for M positions.

4.3.6 Decentralized CIP algorithm

The linear integer programming (LIP) problem presented in Section 4.3.4 can be solved by a standard solver using the branch-and-cut technique. A variation of branch-and-cut also exists for solving the convex IP (CIP) [45]. However, neither of these solution methods allow for a distributed implementation, which is a critical flaw for our decentralized setup. To remedy this problem in the convex case, we develop our own CIP algorithm.[3]

We propose an algorithm which uses the inherently discrete nature of our convex objective function to solve the problem using an approach similar to

[3]A decentralized solution to the linear IP is the subject of future work.

Bisection search. Note that the objective function may only take on values C_{ij} which are elements of the cost matrix. We begin by arranging the entries of C in ascending order, forming an ordered set:

$$C_{i_1 j_1} \leq C_{i_2 j_2} \leq \cdots \leq C_{i_p j_p},\ p \leq MN \tag{4.5}$$

We note that a solution to the original problem with value r corresponds to a feasible point in the following region:

$$\sum S_{ik} T_{ij} f_{ij} \geq R_{jk},\ \forall j, k \tag{4.6a}$$

$$\sum_j f_{ij} \leq 1,\ i = 1, \ldots, n \tag{4.6b}$$

$$f_{ij} \geq 0,\ f_{ij} \in \mathbb{Z} \tag{4.6c}$$

Where T is the threshold matrix defined by:

$$T_{ij} = \begin{cases} 1 & \text{if } C_{ij} \leq r, \\ 0 & \text{otherwise,} \end{cases} \tag{4.7}$$

An objective value $C_{i_l j_l}$ can be achieved if there exists a feasible point in the above region for $r = C_{i_l j_l}$. Feasibility can be checked by solving a Constraint Satisfaction Problem (CSP). If a feasible point is found, we need not consider any costs higher than $C_{i_l j_l}$. If there exists no feasible point, we need not consider any costs lower than $C_{i_l j_l}$. Thus, we are able to do a Bisection search on the ordered set which terminates in $O(\log MN)$ iterations and yields the minimum feasible C_{ij}—the optimum value of our original CIP.

A flowchart illustrating the iterative solution algorithm is given in Fig. 4.3.6.

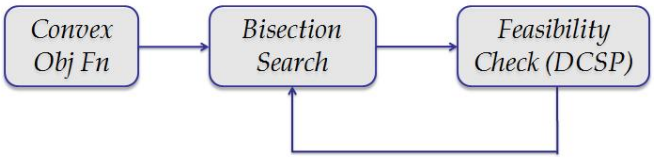


Figure 4.4: A flowchart showing the iterative solution algorithm for the MSRF Assignment problem, comprising a bisection search coupled with a feasibility check.

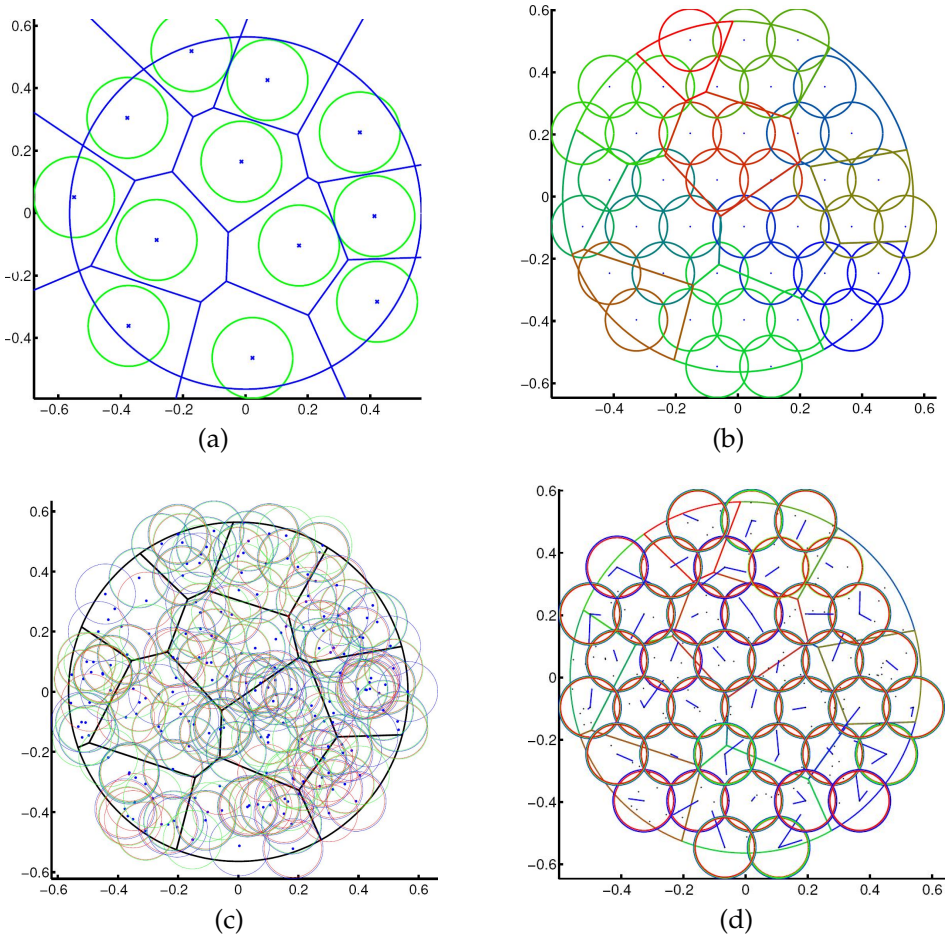


Figure 4.5: Simulation results: (a) shows the cells; (b) presents the optimal grid; (c) depicts the initial state of the nodes; (d) shows their final state.

4.3.7 Distributed Implementation

With careful attention to detail, the above algorithm may be adopted for distributed implementation. To begin, we divide the MSRF into areas of influence which we choose to be the sensing regions SR_b.[4] We cast an entity within each sensing region (an arbitrarily chosen sensor or the beacon itself) as an Autonomous Agent, responsible for assigning sensors to optimal positions within its area of influence.

We proceed to show that these agents can build the equivalent of the ordered cost set without full knowledge of C, proving a convergence result in the process. For simplicity, consider the beacons as agents. First, an arbitrarily rooted minimum spanning tree of agents is constructed. Then, each agent b constructs its own minimum spanning tree including all $i \in SR_b$, with itself as the root node. All sensor nodes pass their positions and capabilities to the agent. The agent now has position, capability, and requirement information about every node i and optimal position j within its area or influence.

For node i, consider the set $\{d_{i1}, ...d_{iM}\}$ of distances to each optimal position. Let $\Delta_i = \left\{\left|d_{ij} - d_{ij'}\right| : j \neq j'\right\}$. The quantity $\epsilon_i = \min \Delta_i$ is the minimum difference between the distances from i to any two optimal positions. Let $\left\{j : d_{ij} \leq \rho\right\}$ be the set of reachable optimal positions when the radius of movement for sensor i is restricted to ρ. We make a fundamental observation: for $\rho < \rho' < \rho + \epsilon_i$, $\left\{j : d_{ij} \leq \rho\right\} = \left\{j : d_{ij} \leq \rho'\right\}$, meaning that the set of reachable positions does not change.

[4]Such division is arbitrary. Area of influence can be chosen to be as large at the entire MSRF or as small as a disk of radius ρ around a single optimal position. As the area decreases, the complexity of the internal problem is traded for the amount of inter-agent communication.

Let each agent calculate ϵ_i for each of its entrusted nodes and derive the minimum. The agents pass their minimum values up the agent spanning tree. The root agent thereby obtains $\epsilon_{min} = \min \epsilon_i$, which it passes back to its child nodes. Note that for $\rho < \rho' < \rho + \epsilon_{min}$, we have $\left\{j : d_{ij} \leq \rho\right\} = \left\{j : d_{ij} \leq \rho'\right\}, \forall i$. This means that during a bisection search, if the difference between the largest known infeasible ρ and the smallest feasible ρ is less than ϵ_{min}, no new feasible positions are possible for $\rho_{infeas} < \rho' < \rho_{feas}$, and the algorithm has converged to an optimum solution.

As a result, the agents may conduct a bisection search armed only with the knowledge of their local variables, ϵ_{min}, and an initial radius ρ_{init}. The agents start with ρ_{init} large enough to ensure a feasible solution at the first iteration; ρ_{init} may need to be as large as the diameter of the FoI to ensure completeness, but should be chosen to be as small as possible given prior knowledge of the FoI's topology. For each iteration t, each agent determines its new radius by $\rho(t+1) = \rho(t) \pm \rho_{init}/2^{t-1}$, the sign depending on whether or not a feasible solution has been found for $\rho(t)$. Solution feasibility is determined in a distributed manner—see next section, and all agents converge at the same time.

We note that the actual value of the objective function may be up to ϵ_{min} less than the smallest ρ_{feas}. To determine this value, we must again resort to inter-agent message passing: once the final assignment vector f_{ij} is obtained, each agent calculates $\max C_{ij} f_{ij}$ in its region, then compares it against the values obtained by its leaf nodes and passes the maximum up the tree. The root agent obtains $\max C_{ij} f_{ij}, \forall i$, which is exactly the value of the global objective function.

The autonomous agent framework can also be used in determining whether a feasible solution exists for a particular value of ρ. This is done through the

Distributed CSP (DCSP) framework proposed by Yokoo [47].

4.3.8 Determining solution feasibility

A DCSP setup consists of three elements: M variables $x_1, x_2, \ldots, x_M$ whose values are taken from finite discrete domains $D_1, D_2, \ldots, D_M$, and a set of inter-variable constraints. Each constraint $p_c(x_{c1}, \ldots, x_{cl})$ is defined as a predicate on the Cartesian product $D_{c1} \times \ldots \times D_{cl}$. These variables are distributed among B agents, meaning that some of the constraints are *local* (a single agent has control of all variables involved) while some are distributed among multiple agents. Agents must find a set of values for their entrusted variables which satisfy both local and inter-agent constraints. The latter are resolved through message-passing between the agents.

Yokoo's algorithm uses a modification of the Asynchronous Weak-Commitment Search Algorithm, and is provably complete (i.e. it will find a satisfying assignment if one exists). Due to space constraints we refer the reader to [47] for the algorithm's details. Here, we limit ourselves to mapping the MSRF feasibility problem onto DCSP framework.

For a given value of ρ, each optimal position $j \in SR_b$ is cast as a variable belonging to agent b and taking values in the domain $D_j = \{sensor_i : d(i, j) \leq \rho\}$. Constraints take the form given in (4.6a) and (4.6b), meaning that each position's sensing requirements must be satisfied, and each sensor must be assigned to at most one optimum position. Within each SR_b, the set of positions $\{j' : |d(j, \rho') - d(j, \rho)| \leq \rho, \rho' \neq \rho\}$ is subject to inter-agent constraints, since D_j may contain sensors outside SR_b. This formulation completes the mapping,

and Yokoo's algorithm may now be applied.

4.4 Implementation

In order to test our algorithm, we implemented a complete simulation testbed. We first used the beacon and Voronoi cell framework (see Section 4.2) to generate a circular FoI of unit area containing b sensing regions (Fig. 4.5a). Note that the sensing regions were generated using circle packing, which yielded cells of similar areas. The FoI was overlaid with an optimal position grid (Fig. 4.5b), with each position inheriting the sensing requirements of the regions intersected by a disk of radius r_s around it. A collection of n sensor nodes was then uniformly distributed within the FoI. Fig. 4.5c shows the initial node deployment, with the concentric circles representing each node's sensing capabilities. Each region's K sensing requirements and each node's K sensing capabilities were generated by the discrete uniform $\{0, 1\}$ distribution, creating an 'average-case' problem.

We then implemented a bisection search algorithm coupled with the framework described in Section 4.3.8. To determine for solution feasibility, we implemented an Asynchronous Weak-Commitment Search with Agent Prioritizing (AWC+AP), where each agent (beacon) assigned a priority value to itself instead of its external variables, and generated *nogood*—inability to satisfy constraints p_c— by performing an exhaustive search on its local problem. [5] Though our algorithm was conceptually identical to that described in [47], we needed to account for a number of technical complexities during development. We pro-

[5] This was done due to time constraints—we recommend implementation of Yokoo's superior multi-AWC algorithm to future researchers interested in the problem.

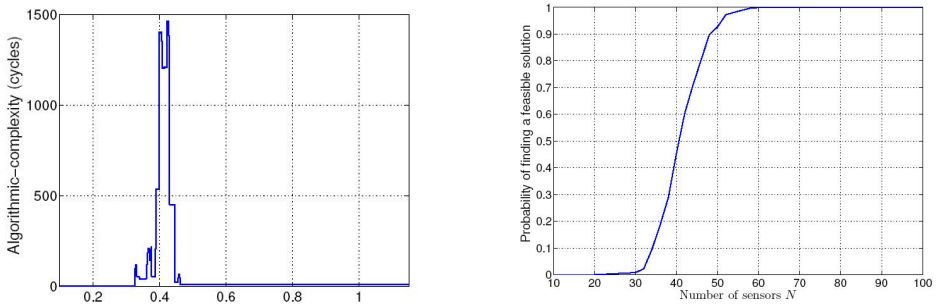
ceed to describe these complexities and our solutions.

The local problem was solved via a branching search with pruning heuristics. Because each position may be assigned more than one sensor, we had to introduce the concept of *supernodes*, which are combinations of up to K sensors. The branching search considered M' optimal positions (meaning that the search tree had a depth of at most M'), cycling through all available supernodes for each position and discarding the ones which resulted in conflict or were unable to fulfill the position's sensing requirements. The addition of each supernode left one or more nodes unavailable for assignment—a list of available nodes, as opposed to available supernodes, was passed down each branch to check for possible conflicts.

Also, it is important to note the structure and nature of the nogood messages. A nogood received by beacon b from beacon b' took the form of an indicator vector and involved all of b''s external sensors—not just the ones shared between b and b'. This was necessary because b is tied to the immediate neighbors of b' via implicit constraints, and a nogood involving only shared sensors would prune off too much of the search space. Again, the reader is referred to [47] for a theoretical explanation.

4.4.1 Results

We tested our algorithm on a wide range of scales and under many differing conditions. There were many options to consider—increasing the number of optimal grid positions, sensing modalities, and/or sensor nodes increased the complexity of the problem, while the ratio of positions to nodes effected both



(a) (b)

Figure 4.6: Experiment results: (a) shows the relative algorithmic-complexity of using convex optimization schemes; (b) shows the probability of finding a feasible solution as a function of the number of sensors deployed in the MSRF.

the amount of required computation and problem feasibility. In this section we comment on the algorithm's performance, describe some of the trends we observed, and make recommendations on effective network design.

We tested two implementations—the distributed CIP algorithm described above and a centralized version, which was essentially the same algorithm run on a single agent. Figures 4.5c and 4.5d show a typical run of our algorithm. The instance shown consisted of 12 sensing regions, 40 optimal positions, 200 sensors, and 3 sensing modalities. The final deployment shown in Fig. 4.5d satisfies all the sensing requirements while maintaining network connectivity. The solid lines in the figure represent the path traveled by each node from its initial position. Note that many nodes (represented by black dots) remain unused. Also note that a single optimal position may be satisfied by more than one sensor.

We gauged algorithm performance by both the amount of calculation and the amount of communication. We chose the calculation metric to be the number of calls to the recursive local search function, since complexity of local search was the dominant component of the calculations. The cost of communication

was counted as both the number of messages exchanged and the number of negotiation rounds until all conflicts were resolved. When comparing the centralized and distributed approaches, the cost of communication was implicit—the more rounds of negotiation between the agents, the more calls to the search function.

Figure 4.6a shows the algorithmic complexity behavior during a typical run. In the figure, there is a clearly defined range of ρ for which the determination of a feasible solution is algorithmically demanding. This region coincides with the feasibility transition region—the region where the problem shifts from infeasible to feasible. In this region, there are very few feasible satisfying solutions and, as a result, it is computationally hard to satisfy all the requirements of the MSRF.

Consequently, a good approximate algorithm is to terminate the bisection search early—this avoids the transition region while giving guaranteed bounds on the error. Changing the ratio of sensors to optimal positions produces a conceptually similar phase transition behavior—for a high enough ratio, a satisfying solution is found every time; for a low enough ratio, the problem instance is almost always infeasible. Fig. 4.6b shows this behavior for a problem instance with the same parameters as those of Fig. 4.6a. For a practical network deployment, it is important to operate comfortably above the transition region, which does not entail uncertainty and high cost of finding a solution.

In all, the performance results were encouraging. Given a sufficiently large node density, the solution was found remarkably quickly, within the first few rounds of inter-agent communication. The performance scaled well with increasing number of positions and sensors, since each agent had a relatively con-

stant number of variables under its control.

Based on our experiments, we conclude that the best operating parameters for a practical deployment are: (1) a sensor to position ratios sufficiently large such that there exist many feasible solutions with high probability; (2) the initial radius ρ_{init} is small enough to confine inter-agent interaction to nearest neighbors; and (3) bisection search is terminated before the DCSP algorithm wanders too far into the high-computation area.

4.5 Conclusions and Future Work

In this chapter, we have formulated a novel class of problems related to mobile wireless networks. We have formalized the problem, introduced the concept of decomposition, and offered a distributed algorithm for performing convex optimization. However, much work remains to be done regarding both theory and implementation.

Future theoretical work should involve developing an algorithm to minimize the linear objective function, possibly by using a multiplier adjustment method. Also, it will be worthwhile to adapt the algorithm to handle infeasible problem instances, developing routines aimed at obtaining the best possible 'nearly-feasible' solutions.

On the practical side, we note that our *CIP* algorithm should be viewed as a benchmark rather than a directly implementable solution. The algorithm yields provably optimal results in terms of coverage, connectivity, and travel time, while making some attempt to minimize the amount of inter-node communica-

tion. However, a 'real-life' reconfiguration algorithm may abandon optimality for reduced communication and simplified implementation.

CHAPTER 5

APPLICATION PLATFORM: ITERATIVELY-DEPLOYED SENSOR NETWORK

5.1 Introduction

In this chapter, we turn toward the practical side of mobile wireless sensor networks, and discuss a project that was done jointly with University of California, Berkeley, as part of its work on NSA's Micro Autonomous Systems and Technology (MAST) initiative [6].

The MAST initiative aims perform "enabling research and transition technology that will enhance warfighter's tactical situational awareness in urban and complex terrain by enabling the autonomous operation of a collaborative ensemble of multifunctional, mobile microsystems" [6]. It includes ten universities and defense contractors, each of which focuses on a particular subproblem, such as microsystem mechanics, microelectronics, and data processing for autonomous operation. The Berkeley team has focused on integration - merging the disparate technologies into cohesive sensing systems - and the work described below is meant to be a demonstration system showcasing what can be done with current capabilities

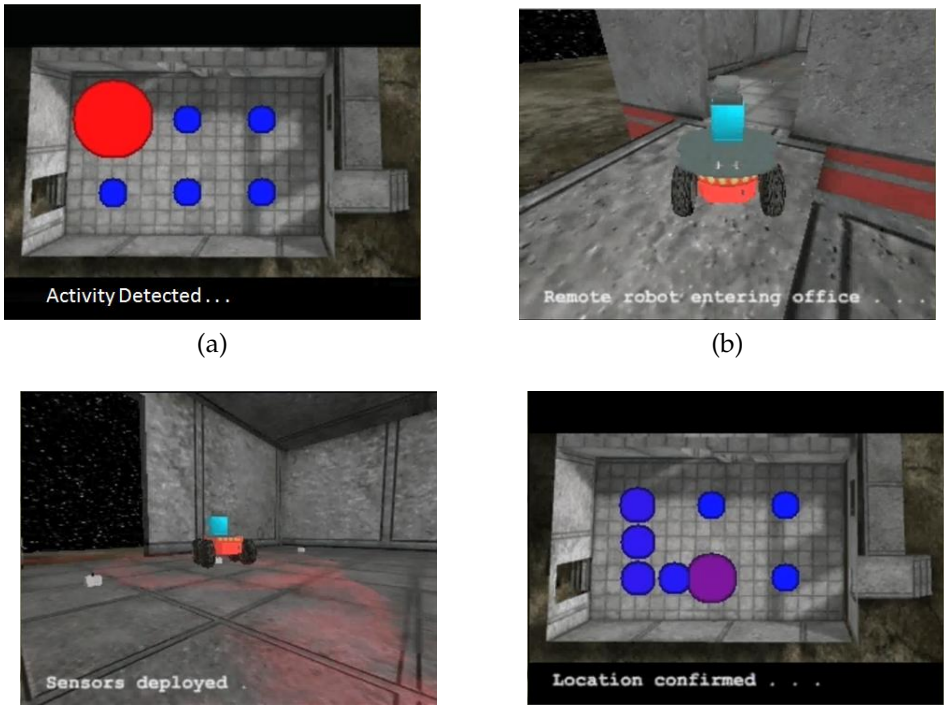
*The work in this chapter was done in collaboration with Hoam Chung, who is a professor at the Department of Mechanical and Aerospace Engineering at Monash University, and Shankar Sastry, who is a professor at the Berkeley Department of Electrical Engineering and Computer Sciences

5.2 Application Scenario

To demonstrate the potential of current sensor networking technology within the context of the MAST initiative, we propose the 'iteratively-deployed wireless network' application scenario. The scenario assumes an initial deployment of a sparse sensor network grid, as shown in Fig. 5.1a. The network consists of stationary wireless network nodes capable of three hundred sixty degree passive infrared (PIR) detection. The network self-organizes upon deployment, and wirelessly transmits any detection event information to an aggregation/data-processing point(a PC, in our case). If any activity is detected by the initial network, the system directs a mobile platform (Fig. 5.1b) to deliver additional nodes (Fig 5.1c), thus increasing the network's resolution at the point of interest (Fig. 5.1d). These additional nodes may be redeployed as necessitated by node supply limitations and evolving sensing needs.

5.3 System Overview

The system as implemented consists of a modified version of the NESTFE demo [7]. The sensor platforms used are Crossbow motes (See Fig. 5.3), which have been modified with additional sensors (we will use only the PIR), small solar arrays and charging circuits, and waterproof enclosures. These motes run a TinyOS program which allows for two-way communication involving both the transmission of detection messages and server-side commands for network configuration. The nodes communicate wirelessly, in single-hop fashion, with a dedicated base-station. The base station uses a Java bridge program to import the data, into Matlab, which then processes the detection messages into a spa-



(a) (b)

(c) (d)

Figure 5.1: Iteratively-Deployed Network application scenario: (a) shows the activity detected by the initial, sparse network; (b) shows a mobile platform moving to the active area; (c) shows the mobile platform deploying additional sensor nodes; (d) shows the resulting denser network.

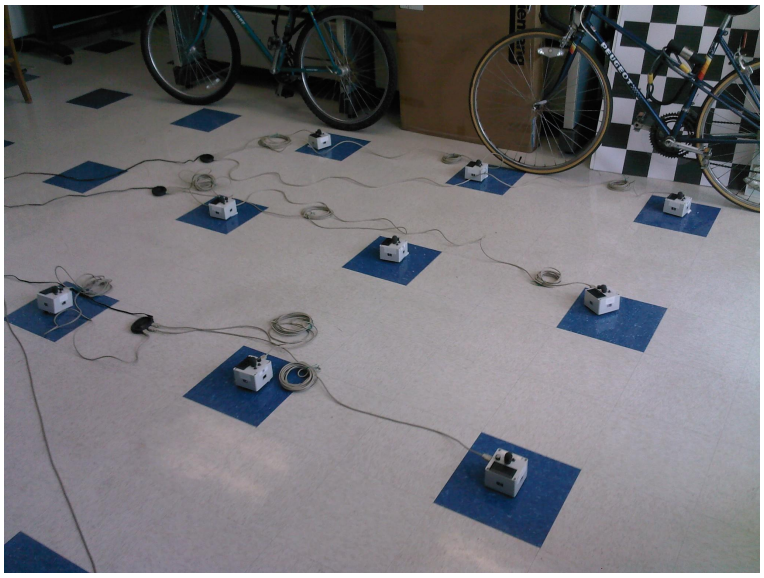


Figure 5.2: Network deployment in a rectangular grid.

tial map and constructs a likelyhood map of target location (Fig. 5.5). This map is communicated to a separate program, which directs the mobile platform to increase network resolution. The mobile deployment platform (not discussed in detail in this chapter) is an autonomous rover modified to carry the motes.

5.4 System Details

5.4.1 Preliminaries

Before beginning work on the demo, one needs to understand the preliminaries of TinyOS as well as the interaction between TinyOS and Matlab. Before beginning, we suggest you read read & understand the following chapters of the TinyOS 1.x tutorial (located on the TinyOS website [46]):

1. Lessons 1-4: Basics

2. Lesson 6: Displaying Data on a PC
3. Matlab: Interacting with motes through Matlab

5.4.2 System Components

The following sections list the component parts of the demo application. They assume possession of the application files, as well as the placement of these files in a specific directory. Included is a brief description of each component, as well as a summary of modifications made, as well as tutorial information. These sections are meant for developers interested in the details of the system's functioning, and may be skipped by casual readers. The system comprises:

Java/NESC 2008DetectDemo:

Builds on the "TestTrioApps" application to implement a single-hop network which reports PIR detections using a simple threshold method. A serverside Java application provides all the functionality of TestTrioApps, along with 1) the ability to set the detection threshold remotely 2) the ability to see the id of the responding node (useful for verification). For an in-depth understanding of this application:

1. Review the TestTrioApps tutorial
2. Review the Java code (CommandWindow.java) contained in C:\Program Files\UCB\cygwin\opt\tinyos-1.x\contrib\nestfe\java\test_trio. Changes from the original TestTrioApps are clearly marked, and include the addi-

tion of a new button/entry field in the PIR tab, as well as code for detection.

3. Revew the DetectionEvent.h , TestTrioMsg.h files contained in C:\Program Files\UCB\cygwin\opt\tinyos-1.x\contrib\nestfe\nesc\apps\ TestTrioApps. These are the two packet formats used for communicating with the nodes. TestTrioMsg.h packets provide for network setup/status reporting, while DetectionEvent.h packets carry information about detection events.
4. Review TestTrioM.nc. Changes from the original TestTrioM are clearly marked, and include implementation of threshold setting/detection as well responding to threshold/PIR Quad/PIR Detect status requests.
5. Be aware of the DetectionEvent.java and TestTrioMsg.java files contained in C:\Program Files\UCB\cygwin\opt\tinyos-1.x\contrib\nestfe\java\ test_trio. These need to be recompiled manually if either of the message formats is changed (see below).

Relevant Compile Commands:

Changing MIG message: Example using TestTrioMsg.h Modify TestTrioMsg.h.h file. In cygwin, enter:

- $mig java -java-classname=test_trio.TestTrioMsg testtriomsg.h TestTrioMsg -o > TestTrioMsg.java
- $javac TestTrioMsg.java

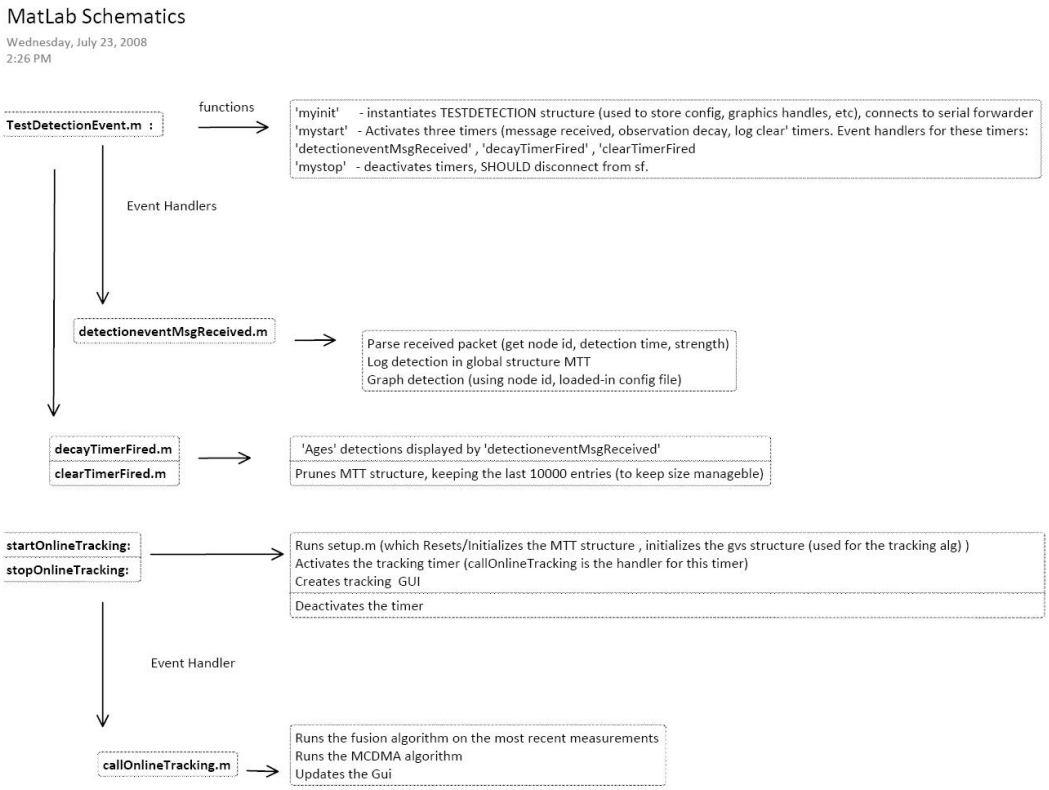


Figure 5.3: File organization flowchart for the Matlab portion of the detection demo.

Matlab 2008DetectDemo:

The demo is found in: C:\Program Files\UCB\cygwin\opt\tinyos1.x\contrib\nestfe\matlab\apps\TestDetectionEvent. It Modifies Songhwai Oh's Detection-Event suite of applications. Fig. 5.4.2 shows the organization of the Matlab modules, their function, and their relationships to one another .Changes to the original application are not always marked, but include:

1. Changing the TestDetectionEvent('myinit')/('mystart') functions to directly receive DetectionEventMsg packets.
2. Changing the detectionEventMsgReceived.m function to handle Detec-

tionEventMsg packets. Note: location info is assigned based on node id within MatLab. Detection time is assigned based on arrival time within MatLab. This works because of the current single-hop nature of the network.

3. Changing how the detectionEventMsgReceived.m function draws detections, provided the 'draw' flag is set to true. The screen no longer resizes with each new detection, providing a more pleasant view. decayTimerFired has been similarly modified.
4. Changing the setup.m function in httpeg - it now includes code to handle current .cfg files (you must include such code, which determines the various setup parameters, if you plan to add new ones). Uncommenting the grid setup routine.
5. Commenting clearTimerFired initialization in TestDetectionEvent('mystart'). A network of our size does not need it.

For an in-depth understanding of this application:

1. Read TinyOS1.x Matlab tutorial
2. Read the Matlab portion of Phoebus's tutorial detection, located on the NESTFE wiki site at http://nest.cs.berkeley.edu/nestfe/ index.php/Detection_Demo
3. Review the function map, provided in Figure 2. It shows, in a hierarchical fashion, the functions involved in importing and processing incoming TestDetectionEvent packets.
4. Review a sample .cfg file. These provide location information about the nodes, based on their node ID's. Note: Currently, your bottom left-hand

coordinate must be zero-zero. Changing this will require changing the way data is handled in fuse.m

5. Know that Java Files Relevant to Matlab reside in: C:\P\rogram Files\UCB\cygwin\opt\tinyos-1.x\tools\java\net\tinyos\drain_msgs\ DetectionEvent, and will need to be manually updated if packet format is changed.

Relevant Cygwin Commands

You may want to put these in your bash script to facilitate program development and execution:

- TTANESC - switches to the Nesc TestTrioApps directory
- TTAJAVA - switches to the java/test_trio directory
- sf - starts serial forwarder. Usage: $sf -comm serial@COMxyz:telos &

Programming Motes: 2008DetectDemo App:

1. $ motelist (to see connected motes and their COM ports)
2. $ TTANESC
3. $ make telosb reinstall,YourMoteID bsl,YourMoteCom

Note: The mote ID's must correspond to those provided in your .cfg file. YourMoteCom is the number of the comm port displayed by 'motelist', minus one.

TOSBase:

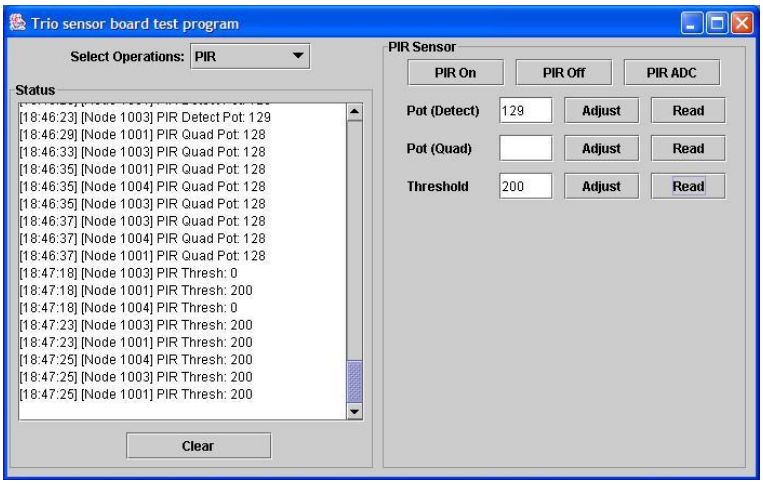


Figure 5.4: Deployment demo network configuration interface.

1. $ cd C:\Program Files\UCB\cygwin\opt\tinyos-1.x\contrib\nestfe\nesc\apps\TOSBase
2. $ make telosb reinstall,YourMoteID bsl,YourMoteCom

5.5 Running The Demo

Start Cygwin. From command line, type:

1. $ TTANESC
2. $ make telosb reinstall [mote num] bsl [com num]
3. $ TTAJAVA
4. $ sf -comm serial@[com num]:telos &
5. $ java test_trio/Injecter 10 &

You should see the serial forwarder window, along with the application window shown in Fig. 5.5.

To operate the network:

1. Select PIR tab.
2. Click PIR on
3. Read the PIR Detect and Quad Potentiometers. Adjust them to a common value (default is 128/128). Make sure all your motes respond - Node ID's are given alongside status messages. Note: because no contention protocol is implemented, for larger networks you will not receive acknowelegement packets from every mote. Press the read button several times to determine that all motes are responding and have the proper value.
4. Set the PIR detection threshold appropriate to the lighting conditions of the room.

Matlab:

Start MatLab. From the command Script, run:

1. >> TestDetectionEvent('myinit')
2. >> TestDetectionEvent('mystart')

If you've set the draw flag to 'true', you will get a real-time picture (See Fig. 5.1a) of the network consisting of nodes and 'detection circles' (which are not sensing radii). The radius of the detection circle corresponds to the strength of the received detection. Line width around the circle corresponds to the detection's freshness.

Once TestDetectionEvent is running, to start the detection / fusion GUI, type:

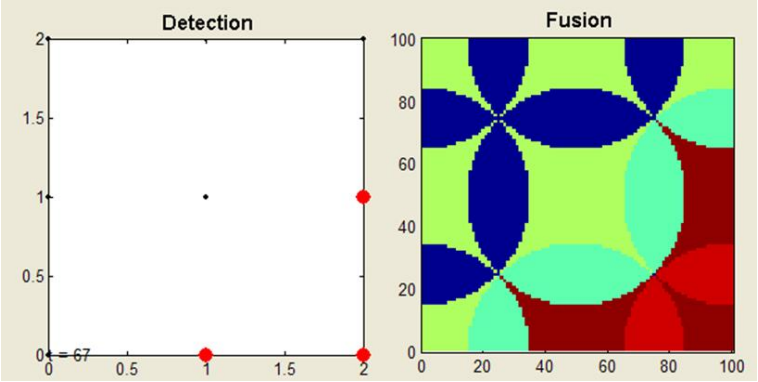


Figure 5.5: System output: The left pane shows detection events as red dots, while the right shows the likelihood map given the detection event history.

>> startOnlineTracking('YourCfgFileName.cfg',0)

To stop, run:

>> stopOnlineTracking

>> TestDetectionEvent('mystop')

You will be shown the GUI in Fig. 5.5. The first window shows detection events as red dots. The second shows the likelihood map, which measures the probability of a target being within a particular area given the detection event history.

CHAPTER 6

CONCLUDING REMARKS

This dissertation has examined several complex problems which dealt with privacy and mobility's impact on wireless sensor network design. Although they may appear to be incongruous issues, mobility and privacy are both highly correlated with each other, and positively correlated with technological enablement. Mobility enables the advancement of wireless sensor networking technology by adding a degree of freedom to the WSN designer's toolkit, making WSNs more easily deployable in any given environment and applicable to a broader range of problems (consequently enabling speedier, more widespread adoption). On the other hand, privacy as an aspect of WSN design works to enable development by tempering potentially invasive technologies, making them conform to social norms (thereby preventing consumer backlash) in the short-term, as well as mitigating more longer-term ethical pitfalls. From the economic perspective, both the commercial success and widespread adoption of WSN technology depend on the public's perception of the degree to which sensing oversteps established privacy boundaries. From the ethical standpoint, even given widespread adoption of sensing systems, individual freedom / capability for self-determination is closely tied to the ability to control the dissemination and use of one's personal information.

We hope that the specific problems presented in this dissertation stimulate further interest in their respective fields, and the presented findings find use with future researchers and practicioners. And while we acknowledge that the individual problems may not fit together direclty, we do believe that they serve the common purpose of enablement, and it is our sincere hope that they

fall together as concrete contributions, however small, in development of next-generation, pervasive wireless sensor networks.

BIBLIOGRAPHY

[1] CERTS Microgrid Test Bed. Website.

[2] Single Point End–Use Energy Disaggregation (SPEED) Marketing Brochure, 2001.

[3] Micro autonomous systems and technology (MAST). collaborative technology alliance (CTA)., July 2006. Draft Program Announcement.

[4] TRUST Website, 2007.

[5] 2008 Assessment of Demand Response and Advanced Metering. Staff Report, dec 2008.

[6] The micro autonomous systems and technology (mast) consortium. Online, apr 2010.

[7] Nest final experiment and maturation of technology. Online, apr 2010.

[8] A. Lee. Privacy and the Law in Demand Response Energy Systems. NIST website, September 2009.

[9] M. G. Lagoudakis A. R. Mosteo, L. Montano. volume 8 of *Distributed Autonomous Robotic Systems-Springer Berlin Heidelberg*, pages 491–502. 2009.

[10] Pramod Abichandani, Hande Y. Benson, and Moshe Kam. Multi-vehicle path coordination in support of communication. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 3839–3846, Piscataway, NJ, USA, 2009. IEEE Press.

[11] Xiaole Bai, Santosh Kumar, Dong Xuan, Ziqiu Yun, and Ten H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proc. of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'06)*, pages 131–142, Florence, Italy, May 22–25, 2006.

[12] Michael Baranski and Voss Jurgen. Detecting Patterns of Appliances from Total Load Data Using a Dynamic Programming Approach. In *Proc. IEEE Fourth International Conference on Data Mining, (ICDM '04)*, pages 327–330, November 2004.

[13] Michael Baranski and Voss Jurgen. Genetic Algorithm for Pattern Detection in NIALM Systems. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3462–3468, oct 2004.

[14] Sarah Bergbreiter and Kristofer S. J. Pister. Design of an autonomous jumping microrobot. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 447–453, Rome, Italy, April 10–14, 2007.

[15] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Multi-agent path planning with multiple tasks and distance constraints. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[16] Matthew Carlyle, Johannes Royset, and Kevin Wood. Lagrangian Relaxation and Enumeration for Solving Constrained Shortest-Path Problems. *Networks*, 52(4):256–270, December 2008.

[17] CPUC. Assigned Comissioner and Administrative Law Judge's Joint Ruling Inviting Comments on Proposed Policies and Findings Pertaining to the Smart Grid Policies Established by the Energy Information and Security Act of 2007. SPUC website, September 2009.

[18] Edsger Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[19] Steven Drenker and Ab Kader. Nonintrusive Monitoring of Electric Loads. In *Proc. IEEE Computer Applications in Power*, volume 12, pages 47–51, dec 1999.

[20] Marshall Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 50(12):1861–1871, dec 2004.

[21] Michael Garey and David Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1979.

[22] Brian P. Gerkey and Maja J. Matarić. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'03)*, volume 3, pages 3862–3868, Taipei, Taiwan, September 14–19, 2003.

[23] H. Nissenbaum. Privacy as Contextual Integrity. *Washington Law Review*, 79(1), 2004.

[24] H. Surden. Structural Rights in Privacy. *SMU Law Review*, 60:1605–1629, 2007.

[25] Eric A. Hansen and Rong Zhou. Anytime heuristic search. *J. Artif. Int. Res.*, 28(1):267–297, 2007.

[26] Nojeong Heo and Pramod K. Varshney. An intelligent deployment and clustering algorithm for a distributed mobile sensor network. In *Proc. IEEE International Conference on Systems, Man and Cybernetics (ICSMC'03)*, volume 5, pages 4576–4581, October 5–8, 2003.

[27] Electric Power Research Institute. Advanced Metering Infrastructure (AMI), feb 2007.

[28] Mark Luk Adrian Perrig Jun Han, Abhishek Jain. Don't Sweat Your Privacy: Using Humidity to Detect Human Presence. In *Proceedings of 5th International Workshop on Privacy in UbiComp (UbiPriv'07)*, September 2007.

[29] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *Proc. of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom'99)*, pages 271–278, Seattle, WA, USA, August 15–19, 1999.

[30] Nejat Karabakal and James Bean. A multiplier adjustment method for multiple shortest path problems with coupling constraints. Technical report, University of Michigan, Ann Arbor, 1995.

[31] Richard E. Korf. Real-time heuristic search. *Artif. Intell.*, 42(2-3):189–211, 1990.

[32] Bhaskar Krishnamachari. *Networking Wirless Sensors*. Cambridge University Press, 2005.

[33] Christopher Laughman, Kwangduk Lee, Robert Cox, Steven Shaw, Steven Leeb, Norford Les, and Peter Armstrong. Power Signature Analysis. *IEEE Power and Energy Magazine*, 1(2):1540–7977, March 2003.

[34] Jack I. Lerner and Deirdre K. Mulligan. Taking the 'Long View' on the Fourth Amendment: Stored Records and the Sanctity of the Home. To be published in the Stanford Technology Law Review, 2008.

[35] Lawrence Lessig. *Code: And Other Laws of Cyberspace, Version 2.0*. Basic Books, New York, NY, second edition, 2006.

[36] Maxim Likhachev. Distributed Path Consensus. 2009.

[37] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. J. Rob. Res.*, 28(8):933–945, 2009.

[38] Maxim Likhachev, Geoffrey J. Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *NIPS*, 2003.

[39] Mulligan, Deirdre K. and Lerner, Jack I. and Jones, Erin and King, Jen and Sislin, Catlin and Wilson, Bethelwel and Hall, Joseph. Privacy and the Law in Demand Response Energy Systems. Samuelson Law, Technology and Public Policy Clinic, 2006.

[40] KXAN Austin News. High Utility Bills May Lead Police To Your Door, nov 2007.

[41] N. J. Nilsson P. E. Hart and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[42] Sameera Poduri and Gaurav S. Sukhatme. Constrained coverage for mobile sensor networks. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'04)*, volume 1, pages 165–172, New Orleans, LA, USA, April 26–May 1, 2004.

[43] Solove, Daniel J. . Understanding Privacy. *Daniel J. Solove, UNDERSTANDING PRIVACY, Harvard University Press, May 2008*.

[44] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[45] Robert A. Stubbs and Sanjay Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532, December 1999.

[46] TinyOS Alliance. Tinyos. Online, apr 2010.

[47] Makoto Yokoo and Katsutoshi Hirayama. Distributed constraint satisfaction algorithm for complex local problems. In *Proc. IEEE International Conference on Multiagent Systems, ICMAS'98)*, pages 372–379, Paris, France, July 3–7, 1998.